А. В. Проскуряков

КОМПЬЮТЕРНЫЕ СЕТИ. Основы построения компьютерных сетей и телекоммуникаций

ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ЮФУ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Инженерно-технологическая академия

А. В. ПРОСКУРЯКОВ

КОМПЬЮТЕРНЫЕ СЕТИ. ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ И ТЕЛЕКОММУНИКАЦИЙ

Учебное пособие

Ростов-на-Дону – Таганрог
Издательство Южного федерального университета
2018

УДК 681.324(07)
ББК 32.973Я73
П824

Печатается по решению кафедры математического обеспечения и применения ЭВМ Института компьютерных технологий и информационной безопасности Южного федерального университета (протокол № 4 от 25 января 2017 г.)

Рецензенты:

кандидат технических наук, доцент, директор направления, руководитель обособленного подразделения ООО «ТЕКТУС.ИТ» в г. Таганроге *Д. П. Калачев*

доктор технических наук, профессор, в.н.с. НИИ МВУС МВС *Н. И. Витиска*

Проскуряков, А. В.

П824 Компьютерные сети. Основы построения компьютерных сетей и телекоммуникаций : учебное пособие / А. В. Проскуряков ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2018. – 201 с.

ISBN 978-5-9275-2792-2

В учебном пособии описаны особенности изучаемого предмета, структура, цели, задачи, основные понятия и общие сведения о компьютерных сетях и сетях передачи данных, эволюция компьютерных сетей, стандартизация в компьютерных сетях, инфраструктура построения сетей, преимущества, требования к компьютерным сетям. В пособии раскрыты примеры топологий, линии связи, кабельные системы, сигналы, кодирование информации, способы и режимы передачи данных, реализация сетевых программных приложений.

Предназначено для студентов направлений подготовки 09.03.04 «Программная инженерия» и 02.03.03 «Математическое обеспечение и администрирование информационных систем», а также может быть использовано студентами всех специальностей направления 09.03.01 «Информатика и вычислительная техника», связанных с использованием ЭВМ, сетевого программного обеспечения при решении прикладных задач, изучающих курс «Компьютерные сети».

УДК 681.324(07)
ББК 32.973Я73

ISBN 978-5-9275-2792-2

ВВЕДЕНИЕ

Данное учебное пособие содержит материал из курса лекций, прочитанных на кафедре МОП ЭВМ для студентов специальностей 230105 «Программное обеспечение вычислительной техники и автоматизированных систем» и 010503 «Математическое обеспечение и администрирование информационных систем», направлений подготовки 23100, 09.03.04 «Программная инженерия», 02.03.03 «Математическое обеспечение и администрирование информационных систем», и может служить для более подробного ознакомления с различными разделами дисциплины.

Согласно учебному плану, данная дисциплина включает: учебных занятий – 180 часов, аудиторных занятий – 72 часа, из них: лекций – 36 часов, лабораторных занятий –36 часов, самостоятельная работа – 72 часа.

Цель дисциплины "Компьютерные сети" состоит в изучении студентами основ и получении знаний в области компьютерных сетей, сетей передачи данных и средств телекоммуникаций, систем обработки данных, архитектуры компьютерных сетей, методов передачи данных, схем кодирования, методов доступа к данным в компьютерных сетях (КС), протоколов и их типовых функций, модели взаимодействия открытых систем, локальных компьютерных сетей, комплекса технических средств и сетевых операционных систем, пакетов электронной почты.

В соответствии с поставленной целью дисциплина изучает:

- системы обработки данных (СОД), сосредоточенных и распределенных, сетевой телеобработки данных, организации компьютерных сетей и сетей передачи данных, структуры КС и СПД;
- элементы теории сигналов, классификацию сигналов, виды модуляции, синхронизацию элементов в сетях;
- методы и способы передачи данных в КС и СПД и схемы кодирования;
- среды передачи данных в КС и СПД, линии связи, каналы связи;
- локальные и удаленные коммуникации, модемы, их основные характеристики и классификации;
- сетевые архитектуры и топологии, применяемые при организации КС и СПД;
- методы доступа к данным в КС и СПД;
- базовую эталонную модель взаимодействия открытых систем, стандарты в области локальных сетей института IEEE;

• протоколы, используемые при взаимодействии прикладных процессов в КС и СПД и их типовых функций;

• локальные компьютерные сети (ЛКС), комплекс технических средств и комплекс программных средств, протоколов, используемых при обмене данными в ЛКС, существующих разновидностей ЛКС;

• типовые пакеты служб электронной почты;

• безопасность передачи и хранения информации в КС и СПД, защиты информации в КС, основных стандартов шифрования при передаче данных.

Специалист должен знать (иметь представление):

• основные понятия систем передачи данных, компьютерных сетей, основные критерии построения компьютерных сетей;

• базовую эталонную модель взаимодействия открытых систем;

• сетевое программное обеспечение, комплекс технических средств;

• протоколы, используемые для взаимодействия в сети;

• безопасность информации при работе в сети;

• стандарты в области локальных компьютерных сетей;

• современные сетевые операционные системы.

Специалист должен уметь:

• работать с утилитами (командами) сетевой операционной системы (Open Enterprise Server – OES, платформа NetWare и SUSE Linux, Windows NT, Linux) в процессе решения и реализации поставленных перед ним практических задач;

• работать с современными пакетами служб электронной почты: P-Mail, Outlook Express, The Bat, Eudora, IncrediMail, Netscape Messenger, Microsoft Outlook;

• применять различные протоколы (IPX/SPX, TCP/IP, FTP, UDP, POP3, SMTP, IMAP, SNMP) при решении поставленных задач;

• применять основные компоненты комплекса технических средств для построения локальных компьютерных сетей и интегрировать их в сети Intranet и Extranet;

• администрировать в сети под управлением СОС (OES-NetWare, Windows NT,Linux).

Конечной целью изучения дисциплины является получение представления:

- о современных сетевых операционных системах (СОС) и их особенностях;
- о современном сетевом оборудовании как комплексе технических средств и возможности его применения при построении компьютерной сети;
- о современных пакетах электронной почты;
- о построении сетей с использованием стандартов (технологий высокой производительности) ATM, ISDN, Frame Relay, SONET/SDH, PDH;
- о построении КС Intranet и Extranet;
- о сетевых службах поддержки работы в сети;
- о перспективах развития сетевых технологий применительно к сетям различного уровня и масштаба.

Основной целью данного пособия является: стремление улучшить доступность материала, увеличить полноту материала, упростить восприятие, отразить основные тенденции в развитии данного направления в целом.

Настоящее учебное пособие является дополнением к учебному пособию «Компьютерные сети и телекоммуникации», часть 1 «Введение в компьютерные сети» лектора общеинститутской дисциплины «Компьютерные сети» профессора кафедры САПР Е. В. Нужнова.

Причины интенсивного развития и роста компьютерных сетей и телекоммуникаций

Мы живем в эпоху бурного и интенсивного развития вычислительных систем. Речь идет не просто о постоянном повышении частоты процессоров, а о том, что конечные станции объединяются сетями передачи данных и превращаются в системы обработки данных. Никому уже не интересна ЭВМ, не имеющая возможности взаимодействия с другими станциями. Что произошло? Ведь на первых этапах гораздо выгоднее было создавать одну большую машину, чем несколько более мелких. Перечислим основные причины, повлекшие за собой рост компьютерных сетей ЭВМ и телекоммуникаций:

- прогресс в области сетей (удешевление и улучшение аппаратной части, а также необходимость в быстрой связи привели к повсеместному внедрению сетей);

- прогресс в области программного обеспечения (появление качественного программного обеспечения для работы и обслуживания сети упрощают работу с ней, тем самым, открывая возможности работы в сети пользователей, не являющихся специалистами в области компьютерных технологий);

- прогресс в области изготовления коммуникационных средств (новые кабельные системы передают информацию дальше и быстрее).

1. КОМПЬЮТЕРНАЯ СЕТЬ. КОМПЬЮТЕРНЫЕ СЕТИ КАК СИСТЕМЫ ОБРАБОТКИ ДАННЫХ

1.1. Системы обработки данных. Классификация систем обработки данных

Начиная изучать ту или иную предметную область, необходимо определиться с терминологией, что особенно важно при изучении такого направления, как компьютерные сети и системы телекоммуникаций. Это особенно важно потому, что данное направление базируется на достижениях таких направлений, как информатика, электротехника, электроника, теория связи, технология материалов, вычислительная техника, вычислительные системы, программное обеспечение (системное и прикладное). Следовательно, необходимо дать определение понятию «компьютерная сеть», а именно «сеть ЭВМ», «вычислительная сеть».

Определение 1. Компьютерная сеть – совокупность сети передачи данных, взаимосвязанных ею ЭВМ и необходимых для реализации этой связи программного обеспечения и технических средств, которая предназначена для организации распределенной обработки информации [1].

Определение 2. Компьютерная сеть – несколько ЭВМ или терминалов, соединенных с помощью одной или большего числа линий связи.

Определение 3. Компьютерная сеть – сеть передачи данных, в одном или нескольких узлах которой размещены ЭВМ.

Каждое из приведенных определений подчеркивает особенность понятия «вычислительная сеть», при этом наиболее полным является определение 1.

В зависимости от сети передачи данных, на базе которой построена компьютерная сеть, различают локальные и территориальные компьютерные сети.

Современные компьютерные сети и системы передачи данных являются системами, причем сложными, и их структурно-физическая и информационно-логическая организация удовлетворяет этому базовому определению.

Современные вычислительные сети и системы передачи данных необходимо рассматривать с точки зрения систем, предназначенных для решения задач обработки информации, т. е. как системы обработки данных (СОД).

Определение 4. Система обработки данных – совокупность комплекса технических и комплекса программных средств, предназначенных для автоматизации и механизации обработки информации.

Системы обработки данных можно классифицировать согласно структурной схеме, приведенной на рис.1. Таким образом, необходимо подчеркнуть, что СОД строятся на базе вычислительных систем и сетей.

Рис. 1. Классификация СОД

Из рис. 1 видно, что имеется два класса СОД:

1) на базе сосредоточенных средств обработки данных (СрОД), в качестве которых выступают отдельные ЭВМ, вычислительные комплексы (многопроцессорные и однопроцессорные), вычислительные системы;

2) на базе распределенных СрОД, в качестве которых выступают системы телеобработки данных и компьютерные сети (от локальных до глобальных).

Компьютерные сети относятся к распределенным (или децентрализованным) территориально вычислительным системам. Поскольку основным признаком распределенной вычислительной системы является наличие нескольких центров обработки данных, то, согласно некоторым классификациям наряду с компьютерными сетями к распределенным системам относят также мультипроцессорные компьютеры и многомашинные вычислительные комплексы, хотя они территориальную распределенность не поддерживают, а данный признак является отличительным для компьютерных сетей.

1.1.1. Мультипроцессорные (многопроцессорные) системы

Определение 5. Мультипроцессорная вычислительная система – это совокупность двух или более взаимосвязанных процессоров, использующих общую память и функционирующих одновременно и согласованно под управлением общей для всех процессоров ОС, которая оперативно распределяет нагрузку между ними [9,10]. Таким образом, в мультипроцессорных системах имеется несколько процессоров, каждый из которых может относительно независимо от остальных выполнять свою программу.

Основные особенности мультипроцессорных систем:

- общая для всех процессоров операционная система, которая оперативно распределяет вычислительную нагрузку между процессорами;
- взаимодействие между отдельными процессорами организуется наиболее простым способом – через общую оперативную память;
- сам по себе процессорный блок не является законченным компьютером и поэтому не может выполнять программы без остальных блоков мультипроцессорного компьютера – памяти и периферийных устройств;
- все периферийные устройства являются общими для всех процессоров мультипроцессорной системы.

Вывод: территориальную распределенность мультипроцессор не поддерживает – все его блоки располагаются в одном или нескольких близко расположенных конструктивах, как и у обычного компьютера.

Достоинства мультипроцессорных систем:

- высокая производительность, которая достигается за счет параллельной работы нескольких процессоров, так как при наличии общей памяти взаимодействие процессоров происходит очень быстро, мультипроцессоры могут эффективно выполнять даже приложения с высокой степенью связи по данным;
- отказоустойчивость (надежность), т. е. способность к продолжению работы при отказах некоторых элементов, например процессоров или блоков памяти, при этом производительность, естественно, снижается, но не до нуля, как в обычных системах, в которых отсутствует избыточность [8, 11, 12, 22].

1.1.2. Многомашинные системы

Многомашинная система – это вычислительный комплекс, включающий в себя несколько компьютеров (каждый из которых работает под

управлением собственной операционной системы), а также программные и аппаратные средства связи компьютеров, которые обеспечивают работу всех компьютеров комплекса как единого целого.

Работа любой многомашинной системы определяется двумя главными компонентами: высокоскоростным механизмом связи процессоров и системным программным обеспечением, которое предоставляет пользователям и приложениям прозрачный доступ к ресурсам всех компьютеров, входящих в комплекс. В состав средств связи входят программные модули, которые занимаются распределением вычислительной нагрузки, синхронизацией вычислений и реконфигурацией системы. Если происходит отказ одного из компьютеров комплекса, его задачи могут быть автоматически переназначены и выполнены на другом компьютере. Если в состав многомашинной системы входят несколько контроллеров внешних устройств, то в случае отказа одного из них, другие контроллеры автоматически подхватывают его работу. Таким образом, достигается высокая отказоустойчивость комплекса в целом.

Помимо повышения отказоустойчивости, многомашинные системы позволяют достичь высокой производительности за счет организации параллельных вычислений. По сравнению с мультипроцессорными системами возможности параллельной обработки в многомашинных системах ограничены: эффективность распараллеливания резко снижается, если параллельно выполняемые задачи тесно связаны между собой по данным. Это объясняется тем, что связь между компьютерами многомашинной системы менее тесная, чем между процессорами в мультипроцессорной системе, так как основной обмен данными осуществляется через общие многовходовые периферийные устройства. Говорят, что в отличие от мультипроцессоров, где используются сильные программные и аппаратные связи, в многомашинных системах аппаратные и программные связи между обрабатывающими устройствами являются более слабыми. Территориальная распределенность в многомашинных комплексах не обеспечивается, так как расстояния между компьютерами определяются длиной связи между процессорным блоком и дисковой подсистемой [9, 11, 12, 22] .

1.1.3. Компьютерные сети

В компьютерных сетях программные и аппаратные связи являются еще более слабыми, а автономность обрабатывающих блоков проявляется в

наибольшей степени – основными элементами сети являются стандартные компьютеры, не имеющие ни общих блоков памяти, ни общих периферийных устройств. Связь между компьютерами осуществляется с помощью специальных периферийных устройств – сетевых адаптеров, соединенных относительно протяженными каналами связи. Каждый компьютер работает под управлением собственной операционной системы, а какая-либо «общая» операционная система, распределяющая работу между компьютерами сети, отсутствует. Взаимодействие между компьютерами сети происходит за счет передачи сообщений через сетевые адаптеры и каналы связи. С помощью этих сообщений один компьютер обычно запрашивает доступ к локальным ресурсам другого компьютера. Такими ресурсами могут быть как данные, хранящиеся на диске, так и разнообразные периферийные устройства – принтеры, модемы, факс-аппараты и т. д. Разделение локальных ресурсов каждого компьютера между всеми пользователями сети – основная цель создания вычислительной сети.

Каким же образом сказывается на пользователе тот факт, что его компьютер подключен к сети? Прежде всего, он может пользоваться не только файлами, дисками, принтерами и другими ресурсами своего компьютера, но и аналогичными ресурсами других компьютеров, подключенных к той же сети. Правда, для этого недостаточно снабдить компьютеры сетевыми адаптерами и соединить их кабельной системой. Необходимы еще некоторые добавления к операционным системам этих компьютеров. На компьютерах, ресурсы которых должны быть доступны всем пользователям сети, необходимо добавить модули, которые постоянно будут находиться в режиме ожидания запросов, поступающих по сети от других компьютеров. Обычно такие модули называются программными серверами (server), так как их главная задача – обслуживать (server) запросы на доступ к ресурсам своего компьютера. На компьютерах, пользователи которых хотят получать доступ к ресурсам других компьютеров, также нужно добавить к операционной системе некоторые специальные программные модули, которые должны вырабатывать запросы на доступ к удаленным ресурсам и передавать их по сети на нужный компьютер. Такие модули обычно называют программными клиентами (client). Собственно же сетевые адаптеры и каналы связи решают в сети достаточно простую задачу – они передают сообщения с запросами и ответами от одного компьютера к другому, а ос-

новную работу по организации совместного использования ресурсов выполняют клиентские и серверные части операционных систем.

Пара модулей «клиент – сервер» обеспечивает совместный доступ пользователей к определенному типу ресурсов, например к файлам. В этом случае говорят, что пользователь имеет дело с файловой службой (service). Обычно сетевая операционная система поддерживает несколько видов сетевых служб для своих пользователей – файловую службу, службу печати, службу электронной почты, службу удаленного доступа и т. п.

Термины «клиент» и «сервер» используются не только для обозначения программных модулей, но и компьютеров, подключенных к сети. Если компьютер предоставляет свои ресурсы другим компьютерам сети, то он называется сервером, а если он их потребляет – клиентом. Иногда один и тот же компьютер может одновременно играть роли и сервера, и клиента [9, 10, 22].

1.1.4. Распределенные программы

Сетевые службы всегда представляют собой распределенные программы. Распределенная программа – это программа, которая состоит из нескольких взаимодействующих частей (в приведенном на рис. 2 примере – из двух), причем каждая часть, как правило, выполняется на отдельном компьютере сети.

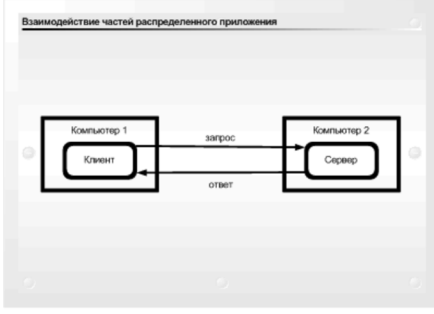

Рис. 2. Распределённая программа

До сих пор речь шла о системных распределенных программах. Однако в сети могут выполняться и распределенные пользовательские программы – приложения. Распределенное приложение также состоит из нескольких частей, каждая из которых выполняет какую-то определенную законченную работу по решению прикладной задачи. Например, одна часть приложения, выполняющаяся на компьютере пользователя, может поддерживать специализированный графический интерфейс, вторая – работать на мощном выделенном компьютере и заниматься статистической обработкой введенных пользователем данных, а третья – заносить полученные результаты в базу данных на компьютере с установленной стандартной СУБД. Распределенные приложения в полной мере используют потенциальные возможности распределенной обработки, предоставляемые вычислительной сетью, и поэтому часто называются сетевыми приложениями.

Следует подчеркнуть, что не всякое приложение, выполняемое в сети, является сетевым. Существует большое количество популярных приложений, которые не являются распределенными и целиком выполняются на одном компьютере сети. Тем не менее, и такие приложения могут использовать преимущества сети за счет встроенных в операционную систему сетевых служб. Значительная часть истории локальных сетей связана как раз с использованием таких нераспределенных приложений.

Большинство приложений, используемых в локальных сетях в середине 80-х гг., являлись обычными, нераспределенными приложениями. И это понятно – они были написаны для автономных компьютеров, а потом просто перенесены в сетевую среду. Создание же распределенных приложений, хотя и сулило много преимуществ (уменьшение сетевого трафика, специализация компьютеров), оказалось делом совсем не простым. Нужно было решать множество дополнительных проблем – на сколько частей разбить приложение, какие функции возложить на каждую часть, как организовать взаимодействие этих частей, чтобы в случае сбоев и отказов оставшиеся части корректно завершали работу и т. д. Поэтому до сих пор только небольшая часть приложений является распределенными, хотя очевидно, что именно за этим классом приложений будущее, так как они в полной мере могут использовать потенциальные возможности сетей по распараллеливанию вычислений [9, 10, 11, 22].

1.2. Основные этапы развития компьютерных сетей как нформационно-вычислительных систем

Развитие информационно-вычислительных систем и сетей и концепция их построения является логическим результатом эволюции компьютерной технологии, прошедшей этапы развития от больших ЭВМ до персональных компьютеров, подключенных к сети, и создания на их основе территориально и функционально распределенных информационно-компьютерных систем и СОД.

1.2.1. Системы пакетной обработки

Фактически системы пакетной обработки являлись системами, в которых отрабатывалась технология оптимизации использования ресурсов отдельно взятой ЭВМ или машинного комплекса и в какой-то мере являлись основой для создания систем распределенной обработки данных. Системы пакетной обработки строились на базе мэйнфрейма – мощного компьютера универсального назначения. Пользователи формировали свои запросы при помощи перфокарт, содержащих данные и команды программ. Затем их передавали в вычислительный центр, где операторы вводили эти карты в компьютер, а на следующий день пользователь получал распечатанные результаты. Таким образом, одна неверно набитая карта означала как минимум суточную задержку. Во главу угла ставилась эффективность работы самого дорогого устройства вычислительной машины – процессора, в ущерб эффективности работы использующих его специалистов. Развитие комплекса технических средств (КТС) вычислительной техники (ВТ) и комплекса программного обеспечения (КПО) привело к увеличению количества пользователей, пользующихся услугами ЭВМ, и разнообразию ЭВМ, что способствовало развитию первого этапа в эволюции компьютерных сетей.

1.2.2. Многотерминальные системы

Этап 1. Многотерминальные информационно-вычислительные системы (ИВС) являются прообразом компьютерных сетей.

Причины появления многотерминальных ИВС:

- удешевление процессоров в начале 60-х гг.;
- появление новых способов организации вычислительного процесса, которые позволили учесть интересы пользователей;
- развитие коммуникационных средств передачи данных.

Следствие. Начали развиваться интерактивные многотерминальные системы разделения времени (рис. 3).

Достоинства многотерминальных ИВС:

- в таких системах компьютер отдавался в распоряжение сразу нескольким пользователям;
- каждый пользователь получал в свое распоряжение терминал, подключенный к центральной ЭВМ, с помощью которого он мог вести с ней диалог;
- время реакции вычислительной системы было достаточно мало для того, чтобы пользователю была не слишком заметна параллельная работа с компьютером и других пользователей, что создавало иллюзию индивидуальной работы и пользования ресурсами ЭВМ;
- разделение машинного времени ЭВМ давало пользователям возможность за сравнительно небольшую плату пользоваться преимуществами компьютеризации.

Вывод. Вычислительные мощности остались централизованными, как и в случае пакетной обработки данных, а отдельные функции «ввода-вывода» распределенными, т. е. были созданы предпосылки создания рассредоточенных информационно-компьютерных систем и СОД на их основе.

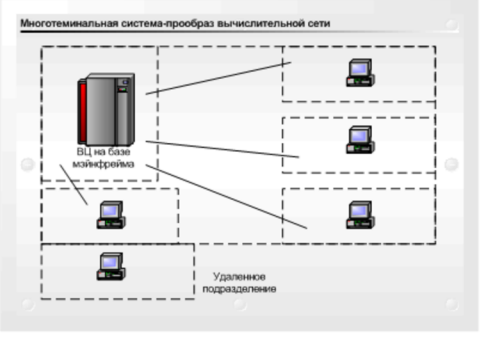

Рис. 3. Многотерминальная система

1.2.3. Глобальные сети

Этап 2. Парадокс развития компьютерных сетей (КС) заключается в том, что первыми появились более широкомасштабные, территориально распределенные КС, вопреки диалектике развития технических систем путем синтеза более сложных систем из более простых: движение от простого к сложному.

Таким образом, назрела потребность в соединении компьютеров, находящихся на большом расстоянии друг от друга, которая заключалась в решении простой задачи – доступа к компьютеру с терминалов, удаленных от него на многие сотни, а то и тысячи километров.

Причины появления глобальных КС (ГКС):

- наличие развитой инфраструктуры в виде телефонных сетей, позволяющих передавать информацию;
- наличие модемов, что позволяло посредством сети передачи данных взаимодействовать с компьютерами;
- необходимость получения многочисленными пользователями удаленного доступа к разделяемым ресурсам нескольких мощных компьютеров класса суперЭВМ.

Следствие. Появились ИВС, в которых наряду с удаленными соединениями типа терминал – ЭВМ были реализованы и удаленные связи типа ЭВМ – ЭВМ. ЭВМ получили возможность обмениваться данными в автоматическом режиме, что является базовым механизмом любой вычислительной сети. На основании этого механизма в первых сетях были реализованы службы обмена файлами, синхронизации баз данных, электронной почты и другие, ставшие теперь традиционными сетевые службы.

При построении глобальных сетей были впервые предложены и отработаны многие основные концепции современных компьютерных сетей. Например:

- многоуровневое построение коммуникационных протоколов;
- технология коммутации пакетов;
- маршрутизация пакетов в составных сетях.

Примером такой сети является сеть Arpanet, из которой выделилась Milnet, а сама Arpanet трансформировалась в Internet.

1.2.4. Первые локальные компьютерные сети

Этап 3. Этот этап можно условно разбить на два подэтапа. Предпосылки появления локальных информационно-компьютерных сетей (ЛИКС) – локальных компьютерных сетей:

- появление в начале 70-х гг. больших интегральных схем (БИС) при их сравнительно невысокой стоимости и высоких функциональных возможностях;
- наличие БИС привело к созданию мини-ЭВМ, которые стали реальными конкурентами мэйнфреймов;
- подразделения предприятий получили возможность приобретать для себя мини-ЭВМ;
- закон Гроша перестал соответствовать действительности, так как десяток мини-ЭВМ выполняли некоторые задачи (как правило, хорошо распараллеливаемые) быстрее одного мэйнфрейма, а стоимость такой мини-компьютерной системы была меньше.

Мини-ЭВМ выполняли задачи управления технологическим оборудованием, складом и другие задачи уровня подразделения предприятия.

Следствие 1. Появилась концепция распределения компьютерных ресурсов по всему предприятию. Однако при этом все ЭВМ одной организации по-прежнему продолжали работать автономно (рис. 4).

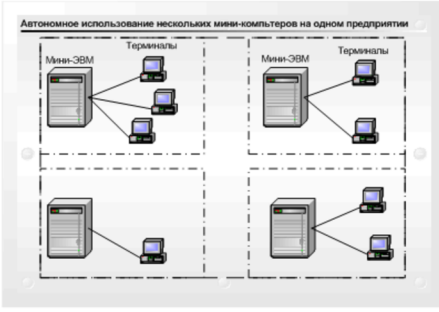

Рис. 4. Автономное использование мини-ЭВМ

Недостатки данной концепции:

- возрастание со временем потребности пользователей в вычислительной технике, так как им стало недостаточно мощности собственных ЭВМ;
- отсутствие возможности обмена данными с другими близко расположенными ЭВМ.

Следствие 2. На предприятиях и организациях стали объединять свои мини-ЭВМ вместе и разрабатывать программное обеспечение, необходимое для их взаимодействия. В результате появились первые локальные компьютерные сети (рис. 5).

Одной из главных проблем было отсутствие стандартов, так как разработчики на отдельных предприятиях разрабатывали свои уникальные устройства сопряжения и ПО для обмена данными между ЭВМ.

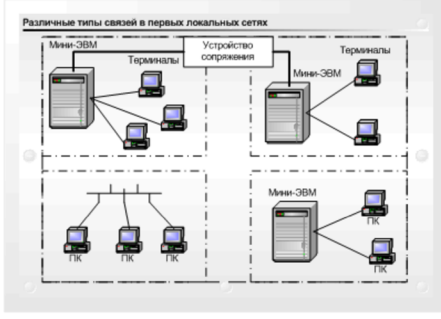

Рис. 5. Объединение мини-ЭВМ в локальную сеть

1.2.5. Стандартные технологии локальных компьютерных сетей

Этап 4. В середине 80-х г г. утвердились стандартные технологии объединения компьютеров в сеть – Ethernet, Arcnet, Token Ring.

Предпосылки появления стандартов ЛКС:

- появление и широкое распространение персональных ЭВМ (ПЭВМ);
- повышение вычислительной мощности ПЭВМ за короткий промежуток времени развития данного класса ЭВМ, что позволило им стать иде-

альными элементами для построения сетей – в качестве серверов (так как они достаточно мощные для работы сетевого программного обеспечения) и в качестве рабочих станций – при условии объединения своей вычислительной мощности для решения сложных задач;

- необходимость разделения дорогих периферийных устройств и дисковых массивов.

Поэтому персональные компьютеры стали преобладать в локальных сетях, причем не только в качестве клиентских компьютеров, но и в качестве центров хранения и обработки данных, т. е. сетевых серверов, потеснив с этих привычных позиций мини-ЭВМ и мэйнфреймы.

Особенности ЛКС:

- локальные сети в сравнении с глобальными сетями внесли много нового в способы организации работы пользователей;
- доступ к разделяемым ресурсам стал гораздо удобнее им – пользователь мог просто просматривать списки имеющихся ресурсов, а не запоминать их идентификаторы или имена;
- возможность реализовать все эти удобства разработчики локальных сетей получили в результате появления качественных кабельных линий связи, на которых даже сетевые адаптеры первого поколения обеспечивали скорость передачи данных до 10 Мбит/с, что значительно повысило производительность ЛКС по сравнению с ГКС со скоростью передачи данных 1200–9600 бит/с;
- повышение надежности;
- реализация различных процедур прозрачного доступа к удаленным ресурсам, стандартным для локальных сетей, для глобальных сетей долго оставались непозволительной роскошью.

1.2.6. Современные тенденции развития компьютерных сетей

В настоящее время компьютерные сети интенсивно развиваются. Основными направлениями развития являются:

- сокращение разрыва между локальными и глобальными сетями (во многом из-за появления высокоскоростных территориальных каналов связи, не уступающих по качеству кабельным системам локальных сетей);
- появление в глобальных сетях служб доступа к ресурсам, таких же удобных и прозрачных, как и службы локальных сетей. Подобные примеры

в большом количестве демонстрирует самая популярная глобальная сеть – Interrnet;

- дальнейшее развитие сетевых протоколов стека TCP/IP, позволяющих получать доступ к информационным ресурсам как ЛКС, так и более широкомасштабных;
- дальнейшая интеграция сетей различного уровня и появление сетей Intranet и Extranet.

Современная сеть Internet объединяет в единое целое многие десятки (а может быть уже и сотни) тысяч локальных сетей по всему миру, построенных на базе самых разных физических и логических протоколов (Fast Ethernet, Gigabit Ethernet, 10Gigabit Ethernet, Token Ring, ISDN, X.25, Frame Relay и т.д.). Эти сети объединяются друг с другом с помощью последовательных каналов (протоколы SLIP, PPP), сетей типа FDDI (часто используется и в локальных сетях), ATM, SDH (Sonet) и многих других. В самих сетях используются протоколы TCP/IP (Интернет), IPX/SPX (Novell), Appletalk, Decnet, Netbios и бесконечное множество других, признанных международными, являющихся фирменными, и т.д. Картина будет неполной, если не отметить многообразие сетевых программных продуктов (Windows NT, NetWare, MultiNet, Lantastic и пр.). На следующем уровне представлены разнообразные внутренние (RIP, IGRP, OSPF) и внешние (BGP, IS-IS и т.д.) протоколы маршрутизации и маршрутной политики, конфигурация сети и задание огромного числа параметров, проблемы диагностики и сетевой безопасности. Немалую трудность может вызвать и выбор прикладных программных средств (Netscape, MS Internet Explorer, Mozilla Firefox, Google Chrom и пр.). В последнее время сети внедряются в управление (CAN), сферу развлечений, торговлю, происходит соединение сетей Internet и кабельного телевидения.

Причиной стремительного роста широкомасштабных сетей явилось то, что создатели базовых протоколов (TCP/IP) заложили в них несколько простых и эффективных принципов:

- инкапсуляцию пакетов;
- фрагментацию/дефрагментацию сообщений;
- динамическую маршрутизацию путей доставки.

Именно эти идеи позволили объединить сети, базирующиеся на самых разных операционных системах (Windows, Unix, Sunos и пр.), использующих различное оборудование (Ethertet, Token Ring, FDDI, ISDN, ATM,

SDH и т.д.), и сделать сеть нечувствительной к локальным отказам аппаратуры. Огромный размер современной сети порождает ряд серьезных проблем. Любое усовершенствование протоколов должно проводиться так, чтобы это не приводило к замене оборудования или программ во всей или даже части сети. Достигается это за счет того, что при установлении связи стороны автоматически выясняют сначала, какие протоколы они поддерживают, и связь реализуется на общем для обеих сторон наиболее современном протоколе (примером может служить использование расширения протокола SMTP – MIME). В кабельном сегменте современной локальной сети можно обнаружить пакеты TCP/IP, IPX/SPX (Novell), Appletalk, которые успешно сосуществуют.

Изменяются и локальные сети. Вместо соединяющего компьютеры пассивного кабеля в них в большом количестве появилось разнообразное коммуникационное оборудование – коммутаторы, маршрутизаторы, межсетевые мосты, шлюзы. Благодаря такому оборудованию появилась возможность построения больших корпоративных сетей, насчитывающих тысячи компьютеров и имеющих сложную структуру. Возродился интерес к крупным компьютерам – в основном из-за того, что после спада эйфории по поводу легкости работы с персональными компьютерами выяснилось, что системы, состоящие из сотен серверов, обслуживать сложнее, чем несколько больших компьютеров. Поэтому на новом витке эволюционной спирали мэйнфреймы стали возвращаться в корпоративные вычислительные системы, но уже как полноправные сетевые узлы, поддерживающие Ethernet или Token Ring, а также стек протоколов TCP/IP, ставший благодаря Internet сетевым стандартом де-факто.

Каждая из сетей, составляющих Internet, может быть реализована на разных принципах. Это может быть Ethernet (наиболее популярное оборудование), Token Ring (вторая по популярности сеть), ISDN, X.25, FDDI или Arcnet. Все внешние связи локальной сети осуществляются через порты-маршрутизаторы (R). Если в локальной сети использованы сети с разными протоколами на физическом уровне, они объединяются через специальные шлюзы (например, Ethernet-Fast Ethernet, Fast Ethernet- Gigabit Ethernet, Ethernet-FDDI и т.д.). Выбор топологии связей определяется многими факторами, не последнюю роль играет надежность. Использование современных динамических внешних протоколов маршрутизации, например BGP-4, позволяет автоматически переключаться на один из альтернативных марш-

рутов, если основной внешний канал отказал. Поэтому для обеспечения надежности желательно иметь не менее двух внешних связей. Сеть LAN-6 (рис. 6) при выходе из строя канала R2-R6 окажется изолированной, а узел LAN-7 останется в сети Internet даже после отказа трех внешних каналов. Широкому распространению Internet способствует возможность интегрировать самые разные сети, при построении которых использованы разные аппаратные и программные принципы. Достигается это за счет того, что для подключения к Internet не требуется какого-либо специального оборудования (маршрутизаторы не в счет, ведь это ЭВМ, где программа маршрутизации реализована аппаратно). Некоторые протоколы из набора TCP/IP (ARP, SNMP) стали универсальными и используются в сетях, построенных по совершенно иным принципам [18, 19, 20, 22].

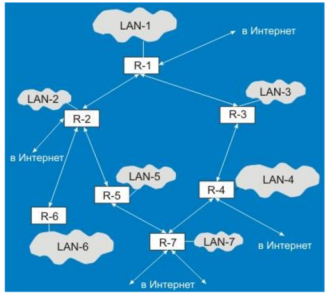

Рис. 6. Схема межсетевых связей

Проявилась еще одна очень важная тенденция, затрагивающая в равной степени как локальные, так и глобальные сети. В них стала обрабатываться несвойственная ранее компьютерным сетям информация – голос, видеоизображения, рисунки. Это потребовало внесения изменений в работу протоколов, сетевых операционных систем и коммуникационного оборудования. Сложность передачи такой мультимедийной информации по сети связана с ее чувствительностью к задержкам при передаче пакетов данных (задержки обычно приводят к искажению такой информации в конечных узлах сети). Так как традиционные службы компьютерных сетей (такие,

как передача файлов или электронная почта) создают малочувствительный к задержкам трафик и все элементы сетей разрабатывались в расчете на него, то появление трафика реального времени привело к большим проблемам.

Сегодня эти проблемы решаются различными способами, в том числе и с помощью специально рассчитанной на передачу различных типов трафика технологии ATM. Однако, несмотря на значительные усилия, предпринимаемые в этом направлении, до приемлемого решения проблемы пока далеко, и в этой области предстоит еще много сделать, чтобы достичь заветной цели – слияния технологий не только локальных и глобальных сетей, но и технологий любых информационных сетей – вычислительных, телефонных, телевизионных и т. д.

Резкое увеличение передаваемых объемов информации в локальных и региональных сетях привело к исчерпанию имеющихся ресурсов, а реальные прогнозы потребностей указывают на продолжение роста потоков в десятки и сотни раз. Единственной технологией, которая способна удовлетворить эти потребности, являются оптоволоконные сети (Sonet, SDH, ATM, FDDI, Fiber Channel). Каналы этих сетей уже сегодня способны обеспечить пропускную способность 155–622 Мбит/с, ведутся разработки и испытания каналов с пропускной способностью в 2–20 раз больше, например гигабитного Ethernet. Осваивается техника мультиплексирования частот в оптоволокне (WDM), что позволяет поднять его широкополосность в 32 раза и в перспективе довести быстродействие каналов до 80 Гбит/с и более. По мере роста пропускной способности возрастают проблемы управления, синхронизации и надежности. Практически все сети строятся сегодня с использованием последовательных каналов. Это связано, прежде всего, со стоимостью кабелей, хотя и здесь существуют исключения (например, HIPPI). Разные сетевые услуги предъявляют разные требования к широкополосности канала. На рис. 7 представлены скоростные диапазоны для основных видов телекоммуникационных услуг. В Internet практически все перечисленные услуги доступны уже сегодня (кроме ТВ высокого разрешения – HDTV).

Рассмотрев диаграмму, можно сделать определенные прогнозы на ближайшее будущее сетей. Через несколько лет можно ожидать слияния функций телевизора и ЭВМ, а это потребует пропускных способностей от магистральных каналов на уровне 0,1–10 Гбит/с. Широкополосность каналов,

приходящих в каждый семейный дом, составляет 1–100 Мбит/с, что позволяет реализовать видеотелефонию, цифровое телевидение высокого разрешения, доступ к централизованным информационным службам и многое другое. Уже существующие оптоволоконные системы обеспечивают в 10 раз большую пропускную способность. Можно предположить и появление локальных сетей внутри жилища. Такие сети способны взять под контроль кондиционирование воздуха, безопасность дома в самом широком смысле этого слова, например оповещение о нежелательном вторжении, пожаре или возможном землетрясении (в сейсмически опасных районах), появление вредных примесей в воздухе. Такая система разбудит хозяина в указанное время, подогреет завтрак, напомнит о предстоящих делах на день, запросит и предоставит хозяину свежий прогноз погоды и справку о состоянии дорог, своевременно сделает заказ на авиабилет и т.д. Все это технологически возможно уже сегодня, пока относительно дорого, но цены весьма быстро падают. Примером может служить сеть CAN, разработанная для сбора данных и управления автомобилем. Стремительное расширение сети Internet не имеет аналогов в истории, так что любой самый фантастический прогноз в этой области может сбыться.

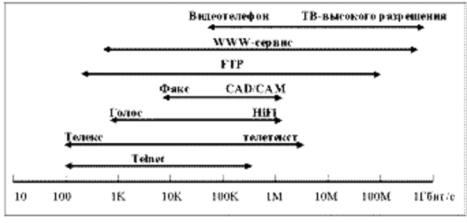

Рис. 7. Диапазоны скоростей телекоммуникационных услуг

Протоколы Internet (TCP/IP) существуют уже около 50 лет. Требования к телекоммуникационным каналам и услугам выросли, и этот набор протоколов не удовлетворяет современным требованиям. Появляются новые протоколы Delta-t (для управления соединением), NetBLT (для передачи больших объемов данных), VMTP (для транзакций; RFC-1045) и XTP (для повышения эффективности передачи данных), блоки протоколов для рабо-

ты с мультимедиа (RTP, RSVP, PIM, ST-II и пр.), но, безусловно, наиболее революционные преобразования вызывает внедрение IPv6.

Хотя сегодня эта идея многим кажется утопией, серьезные специалисты считают, что предпосылки для такого синтеза уже существуют, и их мнения расходятся только в оценке примерных сроков такого объединения – называются сроки от 5 до 10 лет. Причем считается, что основой для объединения послужит технология коммутации пакетов, применяемая сегодня в вычислительных сетях, а не технология коммутации каналов, используемая в телефонии [18, 20, 21, 22].

1.3. Классификация информационно-компьютерных сетей

В настоящее время в технической литературе можно встретить различные схемы классификации компьютерных сетей как разновидности информационно-вычислительных систем. Каждая из этих схем, претендуя на полноту классификации, имеет как достоинства, так и недостатки. В качестве базовой классификационной схемы предложим схему, которая, опираясь на современный уровень развития сетевых технологий, включает следующие классификационные признаки.

1. По назначению КС подразделяют:

- на управляющие (организационными, административными, технологическими и другими процессами);
- информационные (информационно-поисковые, информационно-справочные и т.д.);
- информационно-расчетные;
- КС обработки документальной информации [5, 8, 23].

2. По типам используемых в сети ЭВМ:

- однородные (гомогенные – одноплатформенные по КТС и КПС) компьютерные сети;
- неоднородные (гетерогенные – многоплатформенные по КТС и КПС) компьютерные сети.

3. По территориальной распределенности (масштабу, протяженности):

- локальные компьютерные сети (ЛКС), Local Area Networks (LAN) – это сети компьютеров, сосредоточенные на небольшой территории в пределах одного помещения, лаборатории, отдела, корпуса или нескольких корпусов зданий, расстояние между крайними узлами или рабочими станциями которых составляет от нескольких десятков метров до нескольких

километров (обычно в радиусе не более 1–5 км). В общем случае локальная сеть представляет собой коммуникационную систему, принадлежащую одной организации. Из-за коротких расстояний в локальных сетях имеется возможность использования относительно дорогих высококачественных линий связи, которые позволяют, применяя простые методы передачи данных, достигать высоких скоростей обмена данными в среднем порядка 100 Мбит/с. В связи с этим услуги, предоставляемые локальными сетями, отличаются широким разнообразием и обычно предусматривают их реализацию в режиме on-line;

• городские компьютерные сети (Гор. КС), Metropolitan Area Networks (MAN). Городские сети (или сети мегаполисов) – (MAN) – это сети, предназначенные для обслуживания территории крупного города – мегаполиса. Они появились сравнительно недавно и являются менее распространенным типом сетей. В то время как локальные сети наилучшим образом подходят для разделения ресурсов на коротких расстояниях и широковещательных передач, а глобальные сети обеспечивают работу на больших расстояниях, но с ограниченной скоростью и небогатым набором услуг, сети мегаполисов занимают некоторое промежуточное положение. Они используют цифровые магистральные линии связи, часто оптоволоконные, со скоростями от 45 Мбит/с и предназначены для связи локальных сетей в масштабах города и соединения локальных сетей с глобальными. Эти сети первоначально были разработаны для передачи данных, но сейчас они поддерживают и такие услуги, как видеоконференции и интегральная передача голоса и текста. Развитие технологии сетей мегаполисов осуществлялось местными телефонными компаниями. Исторически сложилось так, что местные телефонные компании всегда обладали слабыми техническими возможностями и из-за этого не могли привлечь крупных клиентов. Чтобы преодолеть свою отсталость и занять достойное место в мире локальных и глобальных сетей, местные предприятия связи занялись разработкой сетей на основе самых современных технологий, например технологии коммутации ячеек SMDS или ATM. Сети мегаполисов являются общественными сетями, и поэтому их услуги обходятся дешевле, чем построение собственной (частной) сети в пределах города;

• региональные компьютерные сети (РКС) – это сети, предназначенные для объединения локальных сетей в масштабах региона или сетей MAN

между собой, в случае большого мегаполиса в качестве региональной сети может выступать сеть MAN;

- общегосударственные компьютерные сети (ОГКС), Wide Area Network (WAN) – это сети предназначенные для объединения городских и региональных сетей в масштабах государства;
- глобальные компьютерные сети (ГКС), Global Area NetWork (GAN) – сети предназначенные для объединения сетей MAN, WAN в сети общепланетарного масштаба. Примером такой сети является Internet.

Примечание. В технической литературе часто сети WAN рассматриваются как глобальные сети, хотя на самом деле общегосударственные сети имеют место на практике и занимают соответствующую позицию в классификационной схеме.

4. По организации управления:

- сети с централизованным управлением. В общем случае ВС с централизованым управлением имеет центральную ЭВМ, управляющую работой сети. Прикладной процесс центральной ЭВМ организует проведение сеансов, связанных с передачей данных, осуществляет диагностику сети, ведет статистический учет работы сети. Для таких сетей характерно обилие служебной информации и приоритетность подключаемых к каналу передачи данных рабочих станций. Сети с централизованным управлением отличаются простотой обеспечения функций взаимодействия между ЭВМ КС и характеризуются тем, что большая часть информационно-вычислительных ресурсов сосредотачивается в центральной ЭВМ. Применение КС с централизованным управлением целесообразно при небольшом числе абонентских систем в виде узлов и рабочих станций;
- сети с децентрализованным управлением. Более целесообразны при равномерном распределении информационных ресурсов по большому числу узлов и рабочих станций. В таких сетях все функции управления распределены между компонентами сети, при этом для проведения диагностики, сбора статистики и выполнения других административных функций в сети используется специально выделенная система или прикладной процесс.

В настоящее время вводится понятие «распределенное управление в КС», хотя по своей сущности это децентрализованное управление.

5. По организации передачи информации (коммутация и маршрутизация) различают КС:

- с коммутацией каналов;
- коммутацией сообщений;
- коммутацией пакетов, которая предусматривает маршрутизацию.

6. По масштабу производственных подразделений:

- сети рабочих групп – это небольшие сети, включающие до 20–25 ЭВМ, организованы как одноранговые КС, обладают простотой и однородностью и приближаются по своим характеристикам к сетям отделов;
- сети отделов – это сети, используемые сравнительно небольшой группой сотрудников, работающих в одном отделе или одной лаборатории предприятия. Главной целью такой сети является разделение локальных ресурсов, таких как приложения, данные лазерные принтеры и модемы. Данные сети строятся на базе какого- либо одного стандарта (например, 100 VG-AnyLAN, Ethernet, Fast Ethernet, Token Ring) и относятся к сетям с выделенным файловым сервером с сетевой операционной системой (СОС) Novell NetWare или Windows NT, а на клиентских машинах может стоять от Windows NT 5.X или клиент Windows 7,8. Однако в качестве СОС для организации сетей отделов могут использоваться Windows или NetWare Lite;
- сети предприятий – это сети, объединяющие множество КС различных отделов, лабораторий, служб одного предприятия или учреждения в пределах отдельного здания или в пределах одной территории, охватывающей площадь в несколько квадратных километров. Основными особенностями таких сетей являются: отсутствие глобальных соединений, предоставление службами сети взаимодействия между сетями отделов, доступ к высокоскоростным модемам и принтерам, доступ к общим базам данных предприятия, доступ к информационным ресурсам сетей других отделов, доступ к корпоративным базам данных. Однако здесь возникают проблемы, связанные с интеграцией неоднородного КТС и КПС, т.е. гетерогенностью, так как платформы ЭВМ, СОС, сетевого аппаратного обеспечения могут отличаться в каждом структурном подразделении и, как следствие, возрастает сложность задачи управления данными сетями;
- корпоративные сети – это сети масштаба предприятия (enterprise-wide networks), имеющие более широкомасштабную территориальную распределенность.

Учитывая специфику современного развития предприятий (они чаще всего территориально распределены, причем территориальная распреде-

ленность может охватывать: город, район, область, регион, государство и даже континент), можно выделить следующие особенности корпоративных сетей:

1) число пользователей и рабочих станций измеряется тысячами;

2) сеть может включать несколько сотен файловых серверов;

3) для организации обмена данными используются глобальные связи;

4) применение разнообразных телекоммуникационных средств, в том числе телефонных каналов, ВОЛС, радиоканалов, спутниковых каналов;

5) гетерогенность сети: использование разнообразных по классам ЭВМ от ПЭВМ до мэйнфрэймов, СОС и сетевых приложений.

Корпоративные сети – пример развития КС, где наглядно проявляется один из базовых философских принципов диалектического развития: переход количества в качество, когда КС предприятия по уровню сложности состава и решаемых задач становится в один ряд с сетями WAN и GAN [9, 10, 13, 22].

1.4. Эффект от использования компьютерных сетей

Применение ИКС позволяет получить следующие преимущества и достигнуть существенного эффекта от совместного использования информационно-вычислительных ресурсов:

1. Сокращение затрат (временных, финансовых) на доступ к информации, интересующей конечного пользователя.

2. Получение доступа к значительным вычислительным мощностям через сеть (доступ к высокопроизводительным и суперЭВМ).

3. Увеличение коэффициента загрузки, за счет уменьшения колебания нагрузки в сети.

4. Доступ к массивам памяти большой емкости (например, Internet).

5. Функциональный набор программного обеспечения, доступ к которому дает компьютерная сеть, конечному пользователю.

6. Доступ к новым информационным технологиям:

- электронная почта;
- телеконференция;
- коммерческие операции, связанные с обслуживанием клиентов через сеть;
- глобальная информационная система.

Конечная цель разработки и применения компьютерных сетей заключается в интегральном обслуживании пользователей.

Контрольные вопросы к разделу 1

1. Перечислите основные причины интенсивного развития систем телекоммуникаций и компьютерных сетей (КС).
2. Дайте определение, что такое компьютерная сеть. Что лежит в основе построения КС?
3. Что такое АСОД? На какие два класса делятся АСОД?
4. На базе каких средств строятся сосредоточенные и распределенные СОД?
5. Что представляют собой мультипроцессорные информационно-вычислительные системы (ИВС)? Перечислите их особенности и основные достоинства.
6. Что представляют собой многомашинные информационно-вычислительные системы (ИВС)? Перечислите их особенности, а также основные достоинства по сравнению с многопроцессорными.
7. Что такое распределенная программа? Основные особенности ее использования применительно к КС.
8. Перечислите основные этапы эволюционного развития КС как ИВС.
9. Охарактеризуйте первый этап развития КС и основные его особенности.
10. Охарактеризуйте второй этап развития КС и основные его особенности.
11. Охарактеризуйте третий этап развития КС и основные его особенности. Следствие данного этапа.
12. Охарактеризуйте четвертый этап развития КС и основные его особенности.
13. Укажите причины более раннего появления глобальных компьютерных сетей (ГКС) по отношению к локальным КС.
14. Стандарты в области ЛКС. Предпосылки появления стандартов в области ЛКС.
15. Перечислите основные особенности ЛКС.
16. Назовите современные тенденции развития КС.

17. В чем заключаются основные причины роста широкомасштабных сетей?
18. Какие сетевые операционные системы (СОС) получили применение в рамках перспективного развития КС?
19. Какие протоколы являются интегрообразующими в КС?
20. Какие основные виды телекоммуникационных услуг предоставляются конечному пользователю в современных КС?
21. Какой вид коммутации имеет наибольшие перспективы применения в КС?
22. Классификация КС. Перечислите основные признаки классификации КС.
23. Приведите классификацию КС по назначению.
24. Приведите классификацию КС по типу используемых ЭВМ.
25. Приведите классификацию КС по территориальной распределенности ЭВМ.
26. Приведите классификацию КС по организации управления.
27. Приведите классификацию КС по организации передачи информации.
28. Приведите классификацию КС по масштабу производственных подразделений.
29. Перечислите, в чем заключается эффект от применения КС.
30. В чем заключается конечная цель разработки и применения КС?

2. ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ

2.1. Требования, предъявляемые к компьютерным сетям. Параметры качества обслуживания в компьютерных сетях

Пользователей различных категорий в первую очередь интересует вопрос, каким требованиям должна удовлетворять КС. Это касается как проектировщиков КС, так и пользователей эксплуатирующих сети. Для оценки выполнения КС основной функции, связанной с обеспечением пользователей потенциальной возможностью доступа к разделяемым ресурсам сети, вводятся параметры качества обслуживания, к которым относятся производительность и надежность. Параметры или требования не ограничиваются только надежностью и производительностью, а включают также управляемость, совместимость, открытость, расширяемость, масштабируемость, прозрачность, поддержку различных видов трафика.

Надежность КС является одним из определяющих требований, предъявляемых к сетям и ИВС, построенным на их базе, так как, если система не обеспечивает необходимой надежности, то смысл пользоваться ее услугами сводится практически к нулю. Одной из первоначальных целей создания распределенных систем являлось достижение большей надежности по сравнению с отдельными вычислительными машинами.

Для технических устройств используются такие показатели надежности, как среднее время наработки на отказ, вероятность отказа, интенсивность отказов. Однако эти показатели пригодны для оценки надежности простых элементов и устройств, которые могут находиться только в двух состояниях – работоспособном или неработоспособном. Сложные системы, состоящие из многих элементов, к которым относятся КС, кроме состояний работоспособности и неработоспособности, могут иметь и другие промежуточные состояния, которые эти характеристики не учитывают. В связи с этим для оценки надежности сложных систем применяется другой набор характеристик.

Отказоустойчивость – способность системы скрыть от пользователя отказ отдельных ее элементов. Например, если копии таблицы базы данных хранятся одновременно на нескольких файловых серверах, то пользователи могут просто не заметить отказ одного из них. В отказоустойчивой системе отказ одного из ее элементов приводит к некоторому снижению качества ее работы (деградации), а не к полному останову. Так, при отказе одного из

файловых серверов в предыдущем примере увеличивается только время доступа к базе данных из-за уменьшения степени распараллеливания запросов, но в целом система будет продолжать выполнять свои функции.

Вероятность доставки пакета узлу назначения без искажений. Так как сеть работает на основе механизма передачи пакетов между конечными узлами, то одной из важных характеристик надежности является вероятность доставки пакета узлу назначения без искажений. Наряду с этой характеристикой могут использоваться и другие показатели: вероятность потери пакета, вероятность искажения отдельного бита передаваемых данных, отношение потерянных пакетов к доставленным. Для контроля за правильностью доставки пакетов вводят контрольные суммы.

Готовность, или коэффициент готовности, означает долю времени, в течение которого система может быть использована. Готовность может быть улучшена путем введения избыточности в структуру системы: ключевые элементы системы должны существовать в нескольких экземплярах, чтобы при отказе одного из них функционирование системы обеспечивали другие.

Непротиворечивость данных. Если для повышения надежности на нескольких файловых серверах хранится несколько копий данных, то нужно постоянно обеспечивать их идентичность.

Безопасность является ключевым требованием общей надежности, т. е. способность системы защитить данные от несанкционированного доступа (НСД). В сетях сообщения передаются по линиям связи, часто проходящим через общедоступные помещения, в которых могут быть установлены средства прослушивания линий. Другим уязвимым местом могут быть оставленные без присмотра персональные компьютеры. Кроме того, всегда имеется потенциальная угроза взлома защиты сети от неавторизованных пользователей, заражение вирусными программами, если сеть имеет выходы в сети MAN, WAN, GAN общего пользования.

Производительность. Существует несколько основных характеристик производительности сети:

- пропускная способность;
- время реакции;
- задержка передачи.

Пропускная способность отражает объем данных, переданных сетью или ее частью в единицу времени. Эта характеристика непосредственно ха-

рактеризует качество выполнения основной функции сети – транспортировки сообщений – и поэтому чаще используется при анализе производительности сети, чем другие характеристики.

Пропускная способность измеряется либо в битах в секунду, либо в пакетах в секунду. Пропускная способность может быть:

средняя – вычисляется путем деления общего объема переданных данных на время их передачи, причем выбирается достаточно длительный промежуток времени – час, день или неделя;

мгновенная – отличается от средней тем, что для усреднения выбирается очень маленький промежуток времени (например, 10 мс или 1 с);

максимальная – наибольшая мгновенная пропускная способность, зафиксированная в течение периода наблюдения.

Время реакции сети определяется как интервал времени между возникновением запроса пользователя к какой-либо сетевой службе и получением ответа на этот запрос.

Время реакции сети обычно складывается из нескольких составляющих. В общем случае в него входят:

- время подготовки запросов на клиентском компьютере;
- время передачи запросов между клиентом и сервером через сегменты сети и промежуточное коммуникационное оборудование;
- время обработки запросов на сервере;
- время передачи ответов от сервера клиенту;
- время обработки получаемых от сервера ответов на клиентском компьютере.

Знание сетевых составляющих времени реакции дает возможность оценить производительность отдельных элементов сети, выявить узкие места и в случае необходимости выполнить модернизацию сети для повышения ее общей производительности.

Задержка передачи определяется как задержка между моментом поступления пакета на вход какого-либо сетевого устройства или части сети и моментом появления его на выходе этого устройства. Этот параметр производительности по смыслу близок ко времени реакции сети, но отличается тем, что всегда характеризует только сетевые этапы обработки данных, без задержек обработки компьютерами сети. Обычно качество сети характеризуют величинами максимальной задержки передачи и вариацией задержки.

Пропускная способность и задержки передачи являются независимыми параметрами, так что сеть может обладать, например, высокой пропускной способностью, но вносить значительные задержки при передаче каждого пакета. Пример такой ситуации дает канал связи, образованный геостационарным спутником. Пропускная способность этого канала может быть весьма высокой, например 2 Мбит/с, в то время как задержка передачи всегда составляет не менее 0,24 с, что определяется скоростью распространения сигнала (около 300 000 км/с) и длиной канала (72 000 км).

Управляемость является одним из важнейших показателей и подразумевает возможность:

- централизованного контроля над состоянием основных элементов сети;
- выявления и разрешения проблем, возникающих в ходе работы сети;
- выполнения анализа производительности и планирование развития сети.

В идеале средства управления сетями представляют собой систему, осуществляющую наблюдение, контроль и управление каждым элементом сети – от простейших до самых сложных устройств, при этом такая система рассматривает сеть как единое целое, а не как разрозненный набор отдельных устройств, то есть с позиций системного подхода.

Расширяемость (extensibility) – возможность добавления отдельных элементов сети (ЭВМ, приложений, служб), наращивания длины сегментов сети и замены существующей аппаратуры более мощной. При этом принципиально важно, что легкость расширения системы иногда может обеспечиваться в некоторых весьма ограниченных пределах. Например, локальная сеть Ethernet, построенная на основе одного сегмента толстого коаксиального кабеля, обладает хорошей расширяемостью, в том смысле, что позволяет легко подключать новые сегменты, хотя и в ограниченном количестве. Однако такая сеть имеет ограничение на число станций – оно не должно превышать 30–40. Хотя сеть допускает физическое подключение к сегменту большего числа станций (до 100), но при этом чаще всего резко снижается производительность сети. Наличие такого ограничения и является признаком плохой масштабируемости системы при хорошей расширяемости.

Масштабируемость (scalability) – возможность сети наращивать количество узлов и протяженность связей в очень широких пределах, при этом производительность сети не ухудшается. Для обеспечения масштабируемо-

сти сети приходится применять дополнительное коммуникационное оборудование и специальным образом структурировать сеть. Например, хорошей масштабируемостью обладает многосегментная сеть, построенная с использованием коммутаторов и маршрутизаторов и имеющая иерархическую структуру связей. Такая сеть может включать несколько тысяч компьютеров и при этом обеспечивать каждому пользователю сети нужное качество обслуживания.

Прозрачность достигается в том случае, когда от пользователя скрывается реальное представление сети и он видит всю сеть как единую вычислительную машину с системой разделения времени. Известный лозунг компании Sun Microsystems: "Сеть – это компьютер" – говорит именно о такой прозрачной сети.

Прозрачность может достигаться на двух разных уровнях – на уровне пользователя и на уровне программиста. На уровне пользователя прозрачность означает, что для работы с удаленными ресурсами он использует те же команды и привычные ему процедуры, что и для работы с локальными ресурсами. На программном уровне прозрачность заключается в том, что приложению для доступа к удаленным ресурсам требуются те же вызовы, что и для доступа к локальным ресурсам. Прозрачность на уровне пользователя достигается проще, так как все особенности процедур, связанные с распределенным характером системы, маскируются от пользователя программистом, который создает приложение. Прозрачность на уровне приложения требует сокрытия всех деталей распределенности средствами сетевой операционной системы.

Поддержка разных видов трафика. Компьютерные сети изначально предназначены для совместного доступа пользователя к ресурсам компьютеров: файлам, принтерам и т.п. Трафик, создаваемый этими традиционными службами компьютерных сетей, имеет свои особенности и существенно отличается от трафика сообщений в телефонных сетях или, например, в сетях кабельного телевидения.

В 90-е гг. компьютерные сети стали использоваться для организации видеоконференций, обучения и развлечения на основе видеофильмов и т.п. Естественно, что для динамической передачи мультимедийного трафика требуются иные алгоритмы и протоколы, и, соответственно, другое оборудование. Хотя доля мультимедийного трафика пока невелика, он уже начал

свое проникновение как в глобальные, так и локальные сети, и этот процесс, очевидно, будет продолжаться с возрастающей скоростью.

Особую сложность представляет совмещение в одной сети традиционного компьютерного и мультимедийного трафиков в силу их отличий. Главной особенностью трафика, образующегося при динамической передаче голоса или изображения, является наличие жестких требований к синхронности передаваемых сообщений, в то время как трафик компьютерных данных характеризуется крайне неравномерной интенсивностью поступления сообщений в сеть при отсутствии жестких требований к синхронности доставки этих сообщений. Для качественного воспроизведения непрерывных процессов, которыми являются звуковые колебания или изменения интенсивности света в видеоизображении, необходимо получение измеренных и закодированных амплитуд сигналов с той же частотой, с которой они были измерены на передающей стороне. При запаздывании сообщений будут наблюдаться искажения.

Сегодня практически все новые протоколы в той или иной степени предоставляют поддержку мультимедийного графика.

Совместимость, или интегрируемость, означает, что сеть способна включать в себя самое разнообразное программное и аппаратное обеспечение. Сеть, состоящая из разнотипных элементов, называется неоднородной или гетерогенной, а если гетерогенная сеть работает без проблем, то она является интегрированной. Основной путь построения интегрированных сетей – выполнение всех модулей в соответствии с открытыми стандартами и спецификациями [5, 9, 10, 22].

2.2. Организация компьютерных сетей. Понятие «сетевая архитектура». Примеры сетевых архитектур

Для объединения отдельных разрозненных ЭВМ в сеть как единую целостную систему необходимо использовать набор принципов и правил, позволяющих выполнить это объединение на структурно-физическом и функционально-логическом уровнях. Структурно-физическая и функционально - логическая организация сети определяется ее архитектурой.

Существует множество разнообразных определений понятия "сетевая архитектура". Приведем некоторые из них:

Определение 1

Сетевая архитектура – это совокупность принципов и правил функционально-логической и структурно-физической реализации как отдельных компонентов сетей телеобработки и компьютерных сетей, так и сетей в целом, построенных из этих компонентов.

Определение 2

Сетевая архитектура – это концепция взаимосвязи элементов сложной структуры. Включает компоненты логической, физической и программных структур.

Определение 3

Архитектура сети – общее описание модели компьютерной сети, определяющей основные ее элементы, характер и топологию их взаимодействия на основе совокупности принципов логической, функциональной и физической организации аппаратных и программных средств [1, 10, 22].

Для сравнения различных видов архитектур необходимо выбрать показатели, лежащие в основе критериев сравнительной оценки различных сетевых архитектур. Такими показателями являются:

- технология соединения (топология);
- производительность;
- надежность;
- программное обеспечение;
- территориальная рассредоточенность (масштабируемость);
- возможность развития (расширяемость, модульность);
- метод доступа к данным [2, 4, 22].

Как видно из вышеуказанных показателей, такой показатель, как топология, определяет структурно-физическую организацию сети.

Примеры зарубежных стандартов сетевых архитектур: DECNET (фирма DEC), SNA (фирма IBM).

В СССР были разработаны аналогичные сетевые архитектуры и средства, получившие название однородных вычислительных сетей. Это архитектура отдельных систем сетевой телеобработки (ОССТ) для ЕС ЭВМ и сетевая архитектура системы малых ЭВМ (САСМ) для СМ ЭВМ.

2.2.1. Топология компьютерной сети. Классификация топологий КС

Топология – это конфигурация соединения компонентов в компьютерной сети.

На топологии как на одном из показателей архитектуры КС акцентируют внимание в большей степени, так как топология определяет многие важные свойства КС: надежность, производительность, живучесть и т.д. Согласно одному из общепринятых подходов, конфигурации КС делятся:

- на широковещательные КС, где каждый приемопередатчик физических сигналов (ЭВМ) передает сигналы, которые могут быть восприняты остальными ЭВМ (общая, или глобальная шина, звезда с пассивным центром, древовидная топология);
- последовательные КС, где каждый физический подуровень передает информацию только одной ЭВМ (звезда с активным центром, кольцевая, иерархическая, последовательная, ячеистая, произвольная).

Широковещательные КС – это сети с селекцией информации, а последовательные – с маршрутизацией.

В настоящее время наиболее широкое распространение получили следующие топологии при реализации КС:

Общая шина (*общая магистраль*). Является очень распространенной (а до недавнего времени самой распространенной) топологией для локальных сетей (рис. 8). В этом случае компьютеры подключаются к одному коаксиальному кабелю по схеме «монтажного ИЛИ». Передаваемая информация может распространяться в обе стороны.

Преимущества: применение общей шины снижает стоимость проводки, унифицирует подключение различных модулей, обеспечивает возможность почти мгновенного широковещательного обращения ко всем станциям сети.

Следовательно, основными преимуществами такой схемы являются дешевизна и простота разводки кабеля по помещениям.

Недостатки: самый серьезный недостаток общей шины заключается в ее низкой надежности – любой дефект кабеля или какого-нибудь из многочисленных разъемов полностью парализует всю сеть. К сожалению, дефект коаксиального разъема не является редкостью. Другим недостатком общей шины является ее невысокая производительность, так как при таком способе подключения в каждый момент времени только один компьютер может передавать данные в сеть. Поэтому пропускная способность канала связи всегда делится здесь между всеми узлами сети. Пример реализации топологии общая шина показан на рис. 8.

Топология «звезда» (рис. 9). В этом случае каждый компьютер подключается отдельным кабелем к общему устройству, называемому концентратором, который находится в центре сети. В функции концентратора входит направление передаваемой компьютером информации одному или всем остальным компьютерам сети. Главное преимущество этой топологии перед общей шиной – существенно большая надежность. Любые неприятности с кабелем касаются лишь того компьютера, к которому этот кабель присоединен, и только неисправность концентратора может вывести из строя всю сеть. Кроме того, концентратор может играть роль интеллектуального фильтра информации, поступающей от узлов в сеть, и при необходимости блокировать запрещенные администратором передачи.

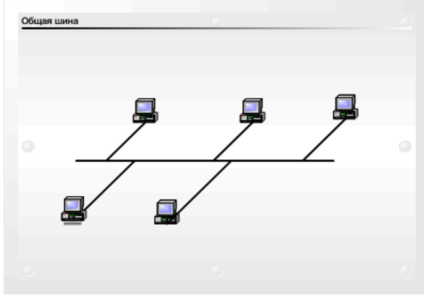

Рис. 8. Топология общая шина

К недостаткам топологии типа «звезда» относится более высокая стоимость сетевого оборудования из-за необходимости приобретения концентратора. Кроме того, возможности по наращиванию количества узлов в сети ограничиваются количеством портов концентратора.

В некоторых случаях строят сеть с использованием нескольких концентраторов, иерархически соединенных между собой связями типа «звезда» (рис. 10). В настоящее время иерархическая звезда является самым распространенным типом топологии связей как в локальных, так и глобальных сетях.

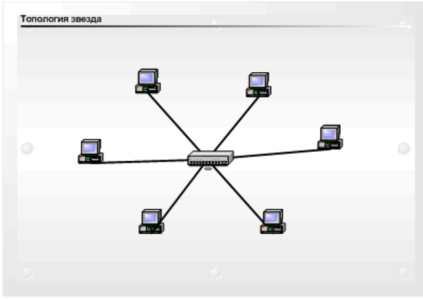

Рис. 9. Топология звезда

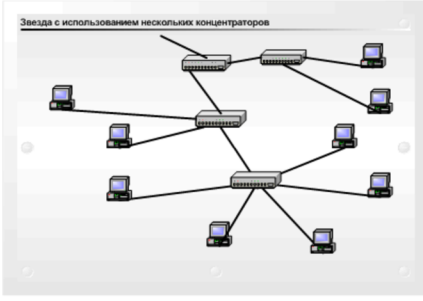

Рис. 10. Топология иерархическая звезда

Топология «звезда» находит применение в таких стандартах ЛКС, как Fast Ethernet, Gigabit Ethernet, Arcnet, в то время как общая шина находит применение в ЛКС стандарта Ethernet.

Кольцевая топология. В сетях с кольцевой конфигурацией (рис.11) данные передаются по кольцу от одного компьютера к другому, как прави-

ло, в одном направлении. Если компьютер распознает данные как «свои», то он копирует их себе во внутренний буфер. В сети с кольцевой топологией необходимо принимать специальные меры, чтобы в случае выхода из строя или отключения какой-либо станции не прервался канал связи между остальными станциями. Кольцо представляет собой очень удобную конфигурацию для организации обратной связи – данные, сделав полный оборот, возвращаются к узлу-источнику. Поэтому этот узел может контролировать процесс доставки данных адресату. Часто это свойство кольца используется для тестирования связности сети и поиска узла, работающего некорректно. Для этого в сеть посылаются специальные тестовые сообщения.

Кольцевая топология находит применение в ЛКС стандарта Token Ring. Данный стандарт находил в некоторое время довольно широкое применение.

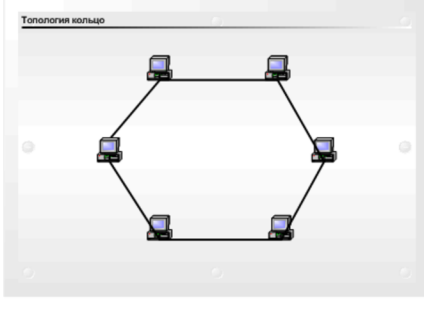

Рис. 11. Кольцевая топология

Полносвязная топология (рис. 12) соответствует сети, в которой каждый компьютер сети связан со всеми остальными. Несмотря на логическую простоту, этот вариант оказывается громоздким и неэффективным. Действительно, каждый компьютер в сети должен иметь большое количество коммуникационных портов, достаточное для связи с каждым из остальных компьютеров сети. Для каждой пары компьютеров должна быть выделена отдельная электрическая линия связи. Полносвязные топологии применя-

ются редко. Чаще этот вид топологии используется в многомашинных комплексах или глобальных сетях при небольшом количестве компьютеров.

Все другие варианты основаны на неполносвязных топологиях, когда для обмена данными между двумя компьютерами может потребоваться промежуточная передача данных через другие узлы сети.

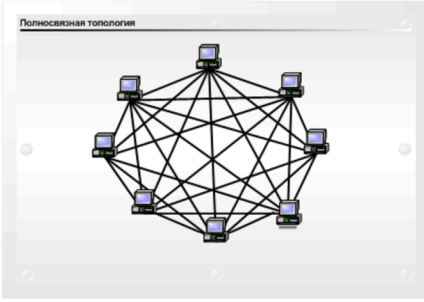

Рис. 12. Полносвязная топология

Ячеистая топология (mesh) получается из полносвязной путем удаления некоторых возможных связей (рис. 13). В сети с ячеистой топологией непосредственно связываются только те компьютеры, между которыми происходит интенсивный обмен данными, а для обмена данными между компьютерами, не соединенными прямыми связями, используются транзитные передачи через промежуточные узлы. Ячеистая топология допускает соединение большого количества компьютеров и характерна, как правило, для глобальных сетей.

В то время как небольшие сети, как правило, имеют типовую топологию – звезда, кольцо или общая шина – для крупных сетей характерно наличие произвольных связей между компьютерами. В таких сетях можно выделить отдельные произвольно связанные фрагменты (подсети), имеющие типовую топологию, поэтому их называют сетями со смешанной топологией (рис. 14).

Компьютеры, подключенные к сети, часто в технической литературе называют рабочими станциями или узлами сети.

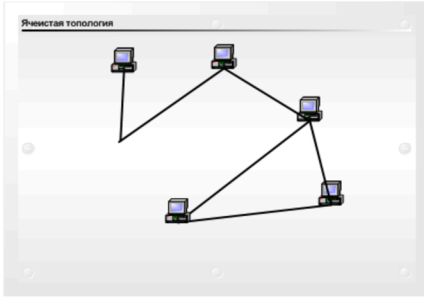

Рис. 13. Ячеистая топология

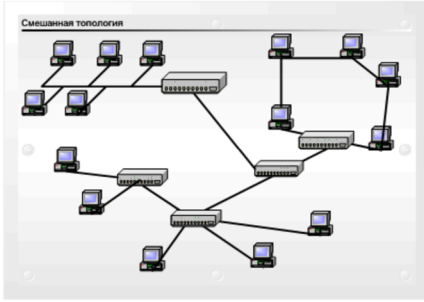

Рис. 14. Смешанная топология

Конфигурация физических связей определяется электрическими соединениями компьютеров между собой и может отличаться от конфигурации логических связей между узлами сети. Логические связи представляют собой маршруты передачи данных между узлами сети и образуются путем соответствующей настройки коммуникационного оборудования.

Вывод. Выбор топологии электрических связей существенно влияет на многие характеристики сети. Наличие резервных связей повышает надежность сети и делает возможным балансирование загрузки отдельных каналов. Простота присоединения новых узлов, свойственная некоторым топологиям, делает сеть легко расширяемой. Экономические требования часто приводят к выбору топологий, для которых характерна минимизация суммарной длины линий связи [6, 9, 13, 22].

2.3. Компоненты компьютерных сетей

КС, как было показано выше, представляет собой сложную систему соответствующим образом структурированную на структурно-физическом и функционально-логическом уровнях. Следовательно, ВС можно представить в виде совокупности базовых составляющих. В основе организации ВС лежит сеть передачи данных (СПД), так как для поддержки распределенной обработки необходимо обеспечивать обмен данными между удаленными ЭВМ. К базовым составляющим ВС, необходимым для ее построения, относятся:

- **линия связи** (рис. 15), состоящая в общем случае из физической среды, по которой передаются электрические информационные сигналы (часто в литературе встречается термин «канал связи»);
- **аппаратура линии связи** – аппаратура передачи данных (АПД или DCE – Data Circuit terminating Equipment), или аппаратура окончания канала данных (АКД), непосредственно связывающая компьютеры или локальные сети пользователя с линией связи;
- **конечное сетевое оборудование**, или оконечное оборудование данных (ООД), являющееся источником и получателем информации, передаваемой по сети. Оно однозначно подпадает под определение DTE (Data Terminal Equipment);
- **коммуникационное сетевое оборудование**, не являющееся источником данных. Вместе с тем ряд устройств, относясь к промежуточным системам, в плане интерфейса может выступать и в роли DCE, и в роли DTE;
- **системное программное обеспечение** – СОС и ОС на серверных и клиентских ЭВМ;
- **сетевые приложения**.

Линия связи (рис.15) состоит в общем случае из физической среды, по которой передаются электрические информационные сигналы. Дополни-

тельно для обеспечения передачи данных в линию связи вводят следующие компоненты: аппаратура передачи данных и промежуточная аппаратура. Синонимом термина «линия связи» (line) в технической литературе является термин «канал связи» (channel), хотя это разные понятия.

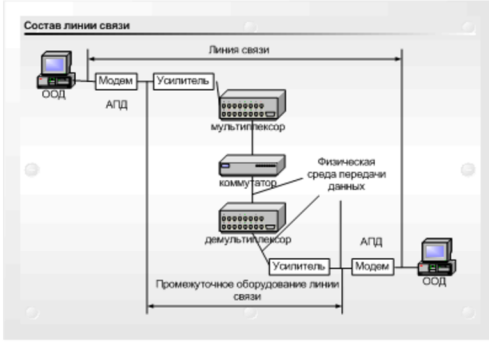

Рис. 15. Состав линии связи

Линия связи – средства, свойства которых обеспечивают распространение сигналов и которые непосредственно используются для соединения между собой АКД, концентраторов данных, коммутаторов и другой аппаратуры, осуществляющей передачу данных.

Канал передачи данных – средства двухстороннего обмена данными, представляющие собой совокупность АКД и линии передачи данных (ЛС).

Физическая среда передачи данных (medium) может представлять собой кабель, т. е. набор проводов, изоляционных и защитных оболочек и соединительных разъемов, а также земную атмосферу или космическое пространство, через которые распространяются электромагнитные волны.

Среда передачи данных – совокупность линий передачи данных и, возможно, коммутаторов, концентраторов, повторителей и другого оборудования, не относящегося к ООД, организация структуры и функционирование которой обеспечивают физическую передачу данных между ООД.

В зависимости от среды передачи данных линии связи разделяются:

- на проводные (воздушные);
- кабельные (медные и волоконно-оптические);
- радиоканалы наземной и спутниковой связи.

Проводные (воздушные) линии связи представляют собой провода без каких- либо изолирующих или экранирующих оплеток, проложенные между столбами и висящие в воздухе. По таким линиям связи традиционно передаются телефонные или телеграфные сигналы, но при отсутствии других возможностей эти линии используются и для передачи компьютерных данных. Скоростные качества и помехозащищенность этих линий оставляют желать много лучшего. Сегодня проводные линии связи быстро вытесняются кабельными.

Кабельные линии представляют собой достаточно сложную конструкцию. Кабель состоит из проводников, заключенных в несколько слоев изоляции: электрической, электромагнитной, механической, а также, возможно, климатической. Кроме того, кабель может быть оснащен разъемами, позволяющими быстро выполнять присоединение к нему различного оборудования. В компьютерных сетях применяются три основных типа кабеля: кабели на основе скрученных пар медных проводов, коаксиальные кабели с медной жилой, а также волоконно-оптические кабели.

Скрученная пара проводов называется витой парой. Витая пара существует в экранированном варианте (STP), когда пара медных проводов обертывается в изоляционный экран, и неэкранированном (UTP), когда изоляционная обертка отсутствует. Скручивание проводов снижает влияние внешних помех на полезные сигналы, передаваемые по кабелю.

Коаксиальный кабель (coaxial) имеет несимметричную конструкцию и состоит из внутренней медной жилы и оплетки, отделенной от жилы слоем изоляции. Существует несколько типов коаксиального кабеля, отличающихся характеристиками и областями применения – для локальных сетей, глобальных сетей, кабельного телевидения и т. п. Волоконно-оптический кабель состоит из тонких (5–60 микрон) волокон, по которым распространяются световые сигналы. Это наиболее качественный тип кабеля – он обеспечивает передачу данных с очень высокой скоростью (до 10 Гбит/с и выше) и к тому же лучше других типов передающей среды обеспечивает защиту данных от внешних помех.

Радиоканалы наземной и спутниковой связи образуются с помощью передатчика и приемника радиоволн. Существует большое количество различных типов радиоканалов, отличающихся как используемым частотным диапазоном, так и дальностью канала. Диапазоны коротких, средних и длинных волн (КВ, СВ и ДВ), называемые также диапазонами амплитудной модуляции (Amplitude Modulation, AM) по типу используемого в них метода модуляции сигнала, обеспечивают дальнюю связь, но при невысокой скорости передачи данных. Более скоростными являются каналы, работающие на диапазонах ультракоротких волн (УКВ), для которых характерна частотная модуляция (Frequency Modulation, FM), а также в диапазонах сверхвысоких частот (СВЧ или microwaves). В диапазоне СВЧ (свыше 4 ГГц) сигналы уже не отражаются ионосферой Земли, и для устойчивой связи требуется наличие прямой видимости между передатчиком и приемником. Поэтому такие частоты используют либо спутниковые каналы, либо радиорелейные каналы, где это условие выполняется.

В компьютерных сетях сегодня применяются практически все описанные типы физических сред передачи данных, но наиболее перспективными являются волоконно-оптические. На них сегодня строятся как магистрали крупных территориальных сетей, так и высокоскоростные линии связи локальных сетей. Популярной средой является также витая пара, которая характеризуется отличным соотношением качества к стоимости, а также простотой монтажа. С помощью витой пары обычно подключают конечных абонентов сетей на расстояниях до 100 метров от концентратора. Спутниковые каналы и радиосвязь используются чаще всего в тех случаях, когда кабельные связи применить нельзя – например, при прохождении канала через малонаселенную местность или же для связи с мобильным пользователем сети.

Следовательно, при проектировании и эксплуатации ВС важное место занимает выбор кабельной системы, так как от этого во многом зависят параметры качества обслуживания [3, 9, 10, 13, 22].

2.3.1. Коммуникационное оборудование

К коммуникационному оборудованию в первую очередь относится АПД. Аппаратура передачи данных (АПД или DCE – Data Circuit terminating Equipment) непосредственно связывает компьютеры или локальные сети пользователя с линией связи и является, таким образом, пограничным оборудованием. Традиционно аппаратуру передачи данных включают в состав

линии связи. Примерами DCE являются модемы, терминальные адаптеры сетей ISDN, оптические модемы, устройства подключения к цифровым каналам. Обычно DCE работает на физическом уровне, отвечая за передачу и прием сигнала нужной формы и мощности в физическую среду.

Аппаратура пользователя линии связи, вырабатывающая данные для передачи по линии связи и подключаемая непосредственно к аппаратуре передачи данных, обобщенно носит название оконечное оборудование данных (ООД или DTE – Data Terminal Equipment). Примером DTE могут служить компьютеры или маршрутизаторы локальных сетей. Эту аппаратуру не включают в состав линии связи.

Разделение оборудования на классы DCE и DTE в локальных сетях является достаточно условным. Например, адаптер локальной сети можно считать как принадлежностью компьютера (DTE), так и составной частью канала связи, т.е. DCE.

Промежуточная аппаратура обычно используется на линиях связи большой протяженности. Промежуточная аппаратура решает две основные задачи: улучшение качества сигнала, создание постоянного составного канала связи между двумя абонентами сети.

В локальных сетях промежуточная аппаратура может совсем не использоваться, если протяженность физической среды – кабелей или радиоэфира – позволяет одному сетевому адаптеру принимать сигналы непосредственно от другого сетевого адаптера, без промежуточного усиления. В противном случае применяются устройства типа повторителей и концентраторов.

В глобальных сетях необходимо обеспечить качественную передачу сигналов на расстояния в сотни и тысячи километров. Поэтому без усилителей сигналов, установленных через определенные расстояния, построить территориальную линию связи невозможно. В глобальной сети необходима также и промежуточная аппаратура другого рода – мультиплексоры, демультиплексоры и коммутаторы. Эта аппаратура решает вторую указанную задачу, то есть создает между двумя абонентами сети составной канал из некоммутируемых отрезков физической среды – кабелей с усилителями. Наличие промежуточной коммутационной аппаратуры избавляет создателей глобальной сети от необходимости прокладывать отдельную кабельную линию для каждой пары соединяемых узлов сети. Вместо этого между мультиплексорами и коммутаторами используется высокоскоростная физическая среда, например волоконно-оптический или коаксиальный кабель,

по которому передаются одновременно данные от большого числа сравнительно низкоскоростных абонентских линий. А когда нужно образовать постоянное соединение между какими-либо двумя конечными узлами сети, находящимися, например, в разных городах, то мультиплексоры, коммутаторы и демультиплексоры настраиваются оператором канала соответствующим образом. Высокоскоростной канал обычно называют уплотненным каналом.

Промежуточная аппаратура канала связи прозрачна для пользователя, он ее не замечает и не учитывает в своей работе. Для него важны только качество полученного канала, влияющее на скорость передачи дискретных данных. В действительности же промежуточная аппаратура образует сложную сеть, которую называют первичной сетью, так как сама по себе она никаких высокоуровневых служб не поддерживает, а только служит основой для построения компьютерных, телефонных или иных сетей.

В зависимости от типа промежуточной аппаратуры все линии связи делятся на:

- аналоговые;
- цифровые.

В аналоговых линиях промежуточная аппаратура предназначена для усиления аналоговых сигналов, т. е. сигналов, которые имеют непрерывный диапазон значений. Такие линии связи традиционно применялись в телефонных сетях для связи АТС между собой. Для создания высокоскоростных каналов, которые мультиплексируют несколько низкоскоростных аналоговых абонентских каналов, при аналоговом подходе обычно используется техника частотного мультиплексирования (FDM).

В цифровых линиях связи передаваемые сигналы имеют конечное число состояний. Как правило, элементарный сигнал, т. е. сигнал, передаваемый за один такт работы передающей аппаратуры, имеет 2 или 3 состояния, которые передаются в линиях связи импульсами прямоугольной формы. С помощью таких сигналов передаются как компьютерные данные, так и оцифрованные речь и изображение. В цифровых каналах связи используется промежуточная аппаратура, которая улучшает форму импульсов и обеспечивает их ресинхронизацию, т.е. восстанавливает период их следования. Промежуточная аппаратура образования высокоскоростных цифровых каналов (мультиплексоры, демультиплексоры, коммутаторы) работает по принципу временного мультиплексирования каналов (TDM), когда каждо-

му низкоскоростному каналу выделяется определенная доля времени (тайм-слот или квант) высокоскоростного канала.

Аппаратура передачи дискретных компьютерных данных по аналоговым и цифровым линиям связи существенно отличается, так как в первом случае линия связи предназначена для передачи сигналов произвольной формы и не предъявляет никаких требований к способу представления единиц и нулей аппаратурой передачи данных, а во втором – все параметры передаваемых линией импульсов стандартизованы. Другими словами, на цифровых линиях связи протокол физического уровня определен, а на аналоговых линиях – нет.

Коммуникационное сетевое оборудование не является источником или конечным получателем данных. Вместе с тем ряд устройств, относясь к промежуточным системам, в плане интерфейса может выступать и в роли DCE, и в роли DTE. Приведем краткие характеристики коммуникационного оборудования локальных сетей [1, 9, 10, 22].

Повторитель (repeater) в сети на коаксиальном кабеле является средством объединения кабельных сегментов в единый логический сегмент (рис. 16). В сетях на витой паре повторитель является самым дешевым средством объединения конечных узлов и других коммуникационных устройств в единый разделяемый сегмент. В этом контексте вместо слова «повторитель» часто применяют «хаб», или «концентратор». Повторители Ethernet могут иметь скорость 10 или 100, 1000 Мбит/с (Fast Ethernet, Gigabit Ethernet), единую для всех портов. Повторитель воспринимает входные импульсы, удаляет шумовые сигналы и передает вновь сформированные пакеты в следующий кабельный сегмент или сегменты. Никакого редактирования или анализа поступающих данных не производится. Задержка сигнала повторителем не должна превышать 7,5 тактов (750 нс для обычного Ethernet). Повторители могут иметь коаксиальные входы/выходы, AUI-разъемы для подключения трансиверов или других аналогичных устройств или каналы для работы со скрученными парами.

Все входы/выходы повторителя с точки зрения пакетов эквивалентны. Если повторитель многовходовый, то пакет, пришедший по любому из входов, будет ретранслирован на все остальные входы/выходы повторителя. Чем больше кабельных сегментов объединено повторителями, тем больше загрузка всех сегментов. При объединении нескольких сегментов с помощью повторителя загрузка каждого из них становится равной сумме

всех загрузок до объединения. Это справедливо как для коаксиальных кабельных сегментов, так и для повторителей, работающих со скрученными парами (хабы – концентраторы). Некоторые повторители контролируют наличие связи между портом и узлом (link status), регистрируют коллизии и затянувшиеся передачи (jabber – узел осуществляет передачу дольше, чем это предусмотрено протоколом), выполняют согласование типа соединения (autonegotiation). В этом случае они обычно снабжены SNMP-поддержкой [19, 20, 22].

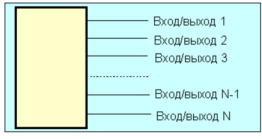

Рис. 16. Сетевой повторитель

Мост (bridge) является средством передачи кадров между двумя (и более) логическими сегментами (рис.17). По логике работы является частным случаем коммутатора. Скорость обычно 10 Мбит/с (для Fast Ethernet 100 Мбит/с используются коммутаторы).

Сетевой мост – это аппаратно-программный блок, который обеспечивает прозрачное соединение нескольких локальных сетей разных стандартов, использующих различные протоколы.

Сетевые мосты делятся на внутренние и внешние:

- внутренние размещаются внутри файлового сервера (ФС);
- внешние – на рабочей станции (РС) вне ФС.

Внешние, в свою очередь, делятся на:

- выделенные – рабочей станцией выполняются функции сетевого моста, работает только как мост;
- невыделенные – рабочей станцией кроме функции сетевого моста могут выполняться другие функции и прикладные задачи.

Мост соединяет два сегмента сети, при инициализации он изучает списки адресов устройств, подсоединенных к каждому из сегментов. В дальнейшем мост записывает в свою память эти списки и пропускает из сегмента в сегмент лишь транзитные пакеты. Существуют мосты, которые оперируют с физическими и с IP-адресами (см. стандарт IEEE 802.1d).

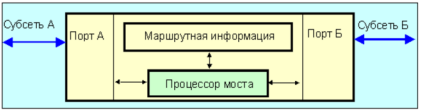

Рис. 17. Межсетевой мост

Мост является активным устройством, которое способно адаптироваться к изменениям в окружающей сетевой среде. При этом пакеты, отправленные из сегмента А и адресованные устройству, которое подключено к этому же сегменту, никогда не попадут в сегмент Б и наоборот. Через мост проходят лишь пакеты, отправленные из сети А в Б или из Б в А.

Функцию моста с определенными скоростными ограничениями может выполнять и обычная ЭВМ, имеющая два сетевых интерфейса и соответствующее программное обеспечение. Мосты при разумном перераспределении серверов и рабочих станций по сетевым сегментам позволяют выровнять и даже эффективно снизить среднюю сетевую загрузку. Когда на один из входов моста приходит пакет, производится сравнение адреса получателя с содержимым внутренней базы данных. Если адрес в базе данных отсутствует, мост посылает широковещательный запрос в порт, противоположный тому, откуда получен данный пакет с целью выяснения местоположения адресата. Понятно, что появление в подсетях А и Б двух объектов с идентичными адресами ни к чему хорошему не приведет. При поступлении отклика вносится соответствующая запись в базу данных. Параллельно анализируется и адрес отправителя и, если этот адрес в базе данных отсутствует, производится его запись в банк адресов соответствующего порта. В базу данных записывается также время записи адреса в базу данных. Содержимое базы данных периодически обновляется. К любой подсети может вести несколько путей, но для нормальной работы мостов и переключателей все пути кроме одного должны быть заблокированы. Функциональная схема работы моста показана на рис. 18. Сети, между которыми включается мост, не обязательно должны работать согласно идентичным протоколам. Возможны мосты между Fast Ethernet и Token Ring или между Fast Ethernet и ATM.

Мост, имеющий более двух портов, называется переключателем.

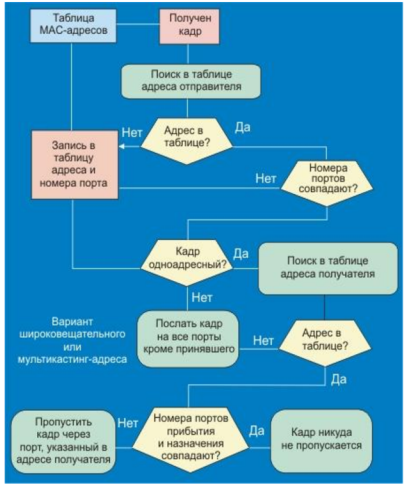

Рис. 18. Алгоритм работы межсетевого моста

Коммутатор (switch) является средством организации виртуальных цепей для передачи каждого кадра между двумя его портами. Скорости портов – 10, 100 или 1000 Мбит/с могут быть разными у разных портов одного устройства. Суммарная полоса пропускания между всеми узлами и коммуникационными устройствами теоретически ограничена производительностью коммутатора. Реальная пропускная способность ниже из-за несимметричности загрузки портов коммутатора. Применяются как средства сегментации сетей, подключения конечных узлов, построения магистралей. Первый переключатель был разработан фирмой Калпане в 1991 году. Иногда переключатели называются маршрутизаторами, тем более что некоторые из них поддерживают внутренние протоколы маршрутизации (например, RIP). Переключатели имеют внутреннюю параллельную магистраль очень высокого быстродействия (от десятков мегабайт до гигабайт в секунду). Эта ма-

гистраль позволяет переключателю совместить преимущества повторителя (быстродействие) и моста (разделение информационных потоков) в одном устройстве. Схемы реализации переключателей варьируются значительно, каких-либо единых стандартов не существует. Алгоритм работы с адресами здесь тот же, что и в случае мостов. На рис.19 приведена схема 8-входового переключателя. В переключателе все входы идентичны, но внешняя информация, записанная в их память, делает входы неэквивалентными. Определенные проблемы возникают, когда к одному из входов переключателя подключен сервер, с которым работают пользователи, подключенные к остальным входам. Если все ЭВМ, подключенные к переключателю, одновременно попытаются обратиться к серверу, переключатель перегрузится и все каналы будут на некоторое время блокированы (будет послан сигнал перегрузки – jam). При данной схеме вероятность таких событий значительна, так как несколько каналов с пропускной способностью 10 Мбит/с работают на один общий канал с той же полосой пропускания. Для преодоления проблем этого рода следует распределять нагрузки между портами переключателя равномерно, а также подключать серверы через дуплексные каналы. Дуплексные каналы полезны и для соединения переключателей между собой. Современные переключатели имеют много различных возможностей – SNMP-поддержка, автоматическая настройка быстродействия и определения типа соединения (дуплексная/полудуплексная). Имеется возможность внешней загрузки программы работы переключателя. Способы проверки производительности переключателей описаны в документах RFC-1242 и RFC-1944 (тесты Бреднера) [19, 20, 21, 22].

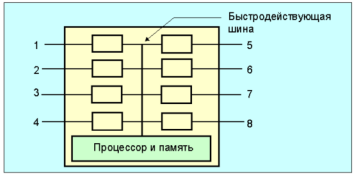

Рис. 19. Переключатель

Существуют переключатели, работающие в режиме “на пролет” (cut through). Здесь первые биты пакета поступают на выход переключателя,

когда последующие еще только приходят на вход. Задержка в этом случае минимальна, но переключатель пропускает через себя пакеты, поврежденные в результате столкновений. Альтернативой такому режиму является передача через буферную память (схема передачи SAF – Store And Forward). Поврежденные пакеты в этом режиме отбрасываются, но задержка заметно возрастает. Кроме того, буферная память (или общая многопортовая) должна иметь место на всех входах. При проектировании сетей следует иметь в виду, что переключатели превосходят маршрутизаторы по соотношению производительность/цена.

При проектировании локальной сети следует учитывать то обстоятельство, что узлы с самым напряженным трафиком должны располагаться как можно ближе к повторителю (мосту или переключателю). В этом случае среднее число коллизий в единицу времени будет ниже. По этой причине сервер должен располагаться как можно ближе к повторителю или другому сетевому устройству (рис. 21).

Схема внутренних связей переключателя может отличаться от приведенной на рис. 19 и иметь конфигурацию, показанную на рис. 20. Привлекательность такой схемы заключается в возможности реализации обмена по двум непересекающимся направлениям одновременно. При этом эффективная пропускная способность многопортового переключателя может в несколько раз превосходить полосу пропускания сети, например 100 Мбит/с.

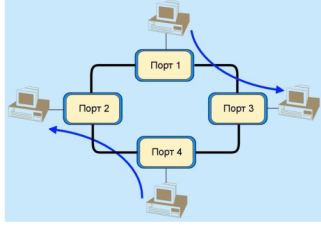

Рис. 20. Конфигурация внутренних связей переключателя

При использовании в сети большого числа мостов и/или переключателей может сформироваться топология связей, когда от одного сегмента к дру-

гому пакет может попасть более чем одним путем (рис. 22). Приведенная на рисунке схема неработоспособна и некоторые связи должны быть ликвидированы. В данном примере проблема может быть решена удалением мостов BR-2 и BR-3 или разрывом связей, помеченных символом "X".

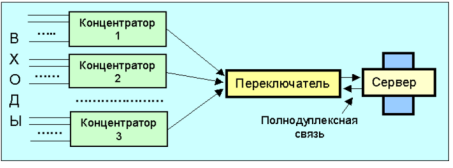

Рис. 21. Подключение переключателя

Проблему ликвидации связей, способных привести к зацикливанию, решает протокол STP (Spanning Tree Protocol; алгоритм предложен Пёлманом в 1992 г.), который автоматически блокирует некоторые соединения, а в случае недоступности основного пути открывает эти заблокированные соединения, обеспечивая высокую надежность сети. STP является частью протокола мостов IEEE 802.1d.

При использовании протокола STP каждой связи присваивается при конфигурации определенный вес (чем меньше, тем выше приоритет). Мосты периодически рассылают специальные сообщения (BPDU – Bridge Protocol Data Unit), которые содержат коды их уникальных идентификаторов, присвоенные им при изготовлении. Мост или переключатель с наименьшим значением такого кода становится корневым ("корень дерева"). Затем выявляется наикратчайшее расстояние от корневого моста/переключателя до любого другого моста в сети. Граф, описывающий дерево наикратчайших связей, и является "расширяющимся деревом". Такое дерево включает все узлы сети, но необязательно все мосты/переключатели. Этот алгоритм функционирует постоянно, отслеживая все топологические изменения.

Современные мосты позволяют создавать виртуальные подсети (VLAN), увеличивающие сетевую безопасность. **VLAN** позволяет ограничить зону распространения широковещательных пакетов, улучшая эксплуатационные характеристики сети в целом.

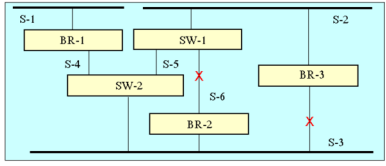

Рис. 22. Подключение межсетевых мостов и переключателей

Некоторые современные мосты используют так называемую маршрутизацию отправителя (source routing). Такая маршрутизация предполагает, что отправитель знает, находится ли адресат в пределах локальной сети, и может оптимально определить путь доставки. При посылке кадра другой сети отправитель устанавливает старший бит своего адреса равным единице. Одновременно в заголовке кадра прописывается весь маршрут. Каждой сети присваивается 12-битовый идентификатор, а каждому мосту ставится в соответствие 4-битовый код, уникальный в контексте данной сети. Это означает, что мосты в пределах одной сети должны иметь разные идентификаторы, но их коды могут совпадать, если они находятся в разных сетях. Мост рассматривает только кадры с единицей в старшем бите адреса места назначения. Для этих кадров просматриваются коды сети в списке, записанном в заголовке. Если в списке содержится код, совпадающий с тем, который характеризует сеть, где находится мост, кадр переадресуется в эту сеть. Реализация алгоритма может осуществляться программно или аппаратно. Если путь до места назначения неизвестен, отправитель генерирует специальный пакет, посылаемый широковещательно (discovery frame) и достигающий всех мостов и всех подсетей. Когда приходит отклик от адресата, мосты записывают его идентификатор, а первичный отправитель фиксирует маршрут до адресата. Данный алгоритм достаточно прост, но сопряжен с лавинным размножением "исследовательских" пакетов, особенно в случае, когда смежные сети соединяются через несколько мостов/переключателей.

Сетевые шлюзы и межсетевые шлюзы. С помощью шлюзов соединяются между собой гетерогенные сети, использующие различные ОС и протоколы высоких уровней. В отличие от сетевых мостов межсетевые шлюзы устраняют несогласованность в скорости передачи данных.

Концентратор (concentrator)**, или хаб** (hub), может включать набор повторителей, коммутаторов и мостов, соединяющих разные стандарты. В технологии FDDI концентраторы используются для организации колец из подключаемых устройств.

Концентратор – устройство, к которому подключаются кабели от множества конечных узлов и коммуникационных устройств. Внутренняя структура может быть различной. Чаще всего под хабом подразумевают повторитель. Сегментирующий хаб (segmented hub, Port Switch hub) является комбинацией нескольких повторителей, между которыми может присутствовать и мост. В сетях Token Ring хаб (MAU, MSAU) является средством организации кольца на физической звездообразной топологии.

Концентраторы (разветвители) HUB бывают:

1) активные (Activ HUB);

2) пассивные (Passiv HUB).

Активные концентраторы, или усилители – это устройства, используемые для усиления сигналов передачи данных в определенных сетевых топологиях.

Как правило, используются для экстенсивно (вширь) - интенсивного (вглубь) наращивания мощности ЛКС.

Пассивный концентратор – устройство, используемое в определенных сетевых топологиях для разделения передаваемых сигналов, что позволяет подключить дополнительные РС.

Преобразователь интерфейсов (media converter) позволяет осуществлять переходы от одной среды передачи к другой (например, от витой пары к оптоволокну) без логического преобразования сигналов. Благодаря усилению сигналов эти устройства могут позволять преодолевать ограничения на длину линий связи (если ограничения не связаны с задержкой распространения). Используются для связи оборудования с разнотипными портами.

Маршрутизатор (router) – устройство с несколькими физическими интерфейсами, возможно, различных сетевых технологий. Выполняет передачи пакетов между интерфейсами на основании информации 3-го (сетевого) уровня. Используется для организации регламентированных связей между логическими сетями (подсетями) на основании сетевой адресной информации, возможно, и с фильтрацией. Узлы сети, желающие переслать пакеты к узлам, не принадлежащим тому же интерфейсу, посылают кадры явно на

MAC – адрес порта маршрутизатора. Маршрутизатор отличается от переключателя тем, что поддерживает хотя бы один протокол маршрутизации. Существуют внутренние и внешние протоколы маршрутизации. Если маршрутизатор осуществляет связь данной автономной системы с другими автономными системами, его называют пограничным (border). Маршрутизатор же, который имеет только один внешний канал связи, в литературе часто называют gateway (входной порт сети). Любой маршрутизатор может поддерживать в любой момент только один внутренний и один внешний протоколы маршрутизации, выбор этих протоколов осуществляет администратор сети из имеющегося списка. Маршрутизаторы представляют собой наиболее сложные сетевые устройства. Главным достоинством маршрутизаторов в локальной сети является ограничение влияния потоков широковещательных сообщений.

В последнее время заметное распространение получил гибрид маршрутизатора и моста – brouter. Некоторые протоколы (например, NetBIOS) не допускают маршрутизации. Когда необходимо использовать такие протоколы совместно с TCP/IP, необходим brouter. Широко используются такие приборы в сетях Token Ring.

Особый класс образуют мультиплексоры/демультиплексоры, которые используют собственные протоколы и служат для предоставления общего канала большему числу потребителей. Эти устройства широко используются при построении сетей типа Интранет (корпоративные сети, где подсети разных филиалов разнесены на большие расстояния). Такие сети строятся на базе специальных выделенных каналов, а мультиплексоры позволяют использовать эти каналы для предоставления комплексных услуг: телефонной связи, передачи факсов и цифровой информации, экономя значительные средства.

Если перед вами стоит задача создания локальной сети с выходом в Интернет, нужно последовательно решить ряд проблем помимо финансовых. Должны быть сформулированы задачи, ради которых эта сеть создается, определена топология сети, число сегментов и характер их связей, число ЭВМ-участников, определен сервис-провайдер, или провайдеры, если вам нужно обеспечить более высокую надежность и живучесть сети. Вам надо оценить требуемую загрузку сегментов сети и внешних каналов связи, выбрать программную среду. После этого вы можете приступить к составлению списка необходимого оборудования и программного обеспечения. Ес-

ли ваша сеть является оконечной и она имеет только один внешний канал связи, вам не нужен маршрутизатор и вы можете ограничиться ЭВМ-портом (gateway), которая должна иметь необходимый интерфейс. Внешним каналом может стать коммутируемая телефонная сеть, выделенная телефонная линия, оптоволоконный кабель или радиорелейный канал. Во всех перечисленных случаях вам будет необходим соответствующий модем.

Брандмауэр (firewall) – устройство (программное средство), по уровню функционирования аналогичное маршрутизатору, но с более развитой системой фильтрации и малым (как правило 2) числом портов. Для сетевых узлов присутствие брандмауэра не должно быть заметно. Используется для защиты локальных сетей от несанкционированного вмешательства извне, обычно устанавливается между маршрутизатором и внешним интерфейсом глобальной сети. Может быть встроен в маршрутизатор или коммуникационное оборудование подключения к глобальной сети.

Модем – устройство (DCE) для передачи данных на дальние расстояния по выделенным или коммутируемым линиям. Интерфейс, обращенный к источнику и приемнику данных (устройству DTE), может быть последовательным (синхронным или асинхронным), параллельным или даже шиной USB.

Модемный пул – сборка из нескольких модемов, которые со стороны, обращенной к DTE, объединены общим портом с интерфейсом ЛВС. Каждый модем пула подключается к своей внешней линии. Устройство позволяет одновременно нескольким (в пределах числа модемов и линий) абонентам локальной сети пользоваться индивидуальными выходами во внешний мир и/или обеспечивать нескольким внешним пользователям доступ к локальной сети.

LAN-модем – комбинация модема и маршрутизатора, имеющая в качестве интерфейса, обращенного к DTE, порт Ethernet (иногда и несколько портов, объединенных повторителем). Позволяет одновременно пользоваться одним выходом во внешний мир группе абонентов локальной сети [19, 20, 21, 22].

2.3.2. Конечное сетевое оборудование

Конечное сетевое оборудование является источником и получателем информации, передаваемой по сети. Оно однозначно подпадает под опреде-

ление DTE (Data Terminal Equipment) и АПД (аппаратура передачи данных).

Компьютер, подключенный к сети, является самым универсальным узлом. Прикладное использование компьютера в сети определяется программным обеспечением и установленным дополнительным оборудованием. Установка мультимедийного оборудования может превратить компьютер в IP-телефон, видеотелефон, терминал видео-конференц-связи. Сетевой интерфейс обеспечивается адаптером локальной сети и программными средствами загружаемой операционной системы. Для дальних коммуникаций используется модем, внутренний или внешний. С точки зрения сети, «лицом» компьютера является его сетевой адаптер. Тип сетевого адаптера должен соответствовать назначению компьютера и его сетевой активности. Компьютеры, подключенные к сети, называются рабочими станциями (PC).

Рабочая станция (WorkStation) – это ПЭВМ, мини-ЭВМ, мэйнфрейм, подключенная к сети посредством сетевого адаптера и используемая для выполнения задач посредством прикладных программ и утилит.

Сервер является тем же компьютером, но подразумевается его более высокая сетевая активность и значимость. Для подключения к сети желательно использовать полнодуплексные высокопроизводительные адаптеры для шин PCI (ISA, EISA, MCA). Серверы желательно подключать к выделенному порту коммутатора в полнодуплексном режиме. При установке двух и более сетевых интерфейсов (в том числе и модемного подключения) и соответствующего ПО сервер может играть роль маршрутизатора или моста. Для повышения пропускной способности сетевого подключения применяют объединение каналов. Конструктивно сервер может представлять собой обычный настольный (desktop) компьютер. Специализированные серверы имеют более просторный конструктив, обеспечивающий большую расширяемость (число процессоров, объем оперативной и дисковой памяти). Серверы могут иметь возможность «горячей» замены (hot swap) дисковых накопителей, резервирование питания, блокировку несанкционированного доступа, средства мониторинга состояния.

Файловый сервер – это ЭВМ, на который установлена сетевая ОС для управления сетью. ФС координирует работу всех PC и регулирует совместное использование сетевых ресурсов этими станциями.

Терминалы, алфавитно-цифровые и графические, используются в клиент-серверных системах в качестве рабочих мест пользователей, а так-

же в качестве консоли для управления сетевым оборудованием. Терминалы, как правило, имеют последовательный интерфейс RS-232C. Терминал имеет собственную систему команд, системы команд различаются в кодировке и трактовке управляющих символов и последовательностей, различаются и таблицы кодировки символов. Терминал может эмулироваться и персональным компьютером, при этом в качестве интерфейса может выступать как COM-порт, так и сетевой интерфейс.

Разделяемые принтеры обеспечивают печать заданий от множества пользователей локальной сети. Для этого требуется сервер печати. **Сервер печати** – средство выборки заданий из очереди (очередей) и, собственно, принтер, логически подключенный к серверу печати. В роли сервера печати может выступать обычный компьютер, подключенный к сети, и принтер подключается к его порту. Использование разделяемых принтеров, особенно лазерных, в графическом режиме с высоким разрешением значительно нагружает сеть. Сервер печати (сетевой принтер) желательно подключать к выделенному порту коммутатора, полный дуплекс ему не нужен.

Сетевые принтеры в дополнение к локальному имеют встроенный сетевой интерфейс Ethernet на 10 или 100 Мбит/с. В этом случае у них должно присутствовать встроенное ПО (сервер печати), рассчитанное на тот или иной сетевой протокол. Сетевой принтер территориально может располагаться в любом месте помещения, где есть розетка кабельной сети.

Разделяемое использование возможно и для плоттеров, если они поддерживают стандартный протокол управления потоком данных. В случае параллельного интерфейса плоттера проблем не возникает – его подключают к любому серверу печати.

Рассмотрим процессы, протекающие в сети, на примере ее фрагмента, показанного на рис. 23, с использованием основных вышеперечисленных понятий.

Рис. 23. Фрагмент компьютерной сети

На ООД1 запущен прикладной процесс ППА1 в точке А1, который взаимодействует с ППВ1 в точке В1, имеющим место в ООД2, через логические связи, обеспечиваемые физически АКД1 и АКД2, и линией связи (ЛС). Посредством прикладного процесса ППВ1 пользователь ООД1 получает доступ к информации, хранящейся в удаленной базе данных. Аналогичным образом на противоположной стороне в ООД2 прикладной процесс ППВ2 посредством взаимодействия с прикладным процессом ППА2 позволяет пользователю получить доступ к определенному файлу БД [1, 22].

2.4. Расширенная структура сети передачи данных. Двухточечное и многоточечное соединения

При организации сетей передачи данных и компьютерных сетей существуют различные технологии соединения оконечного оборудования данных между собой. Устройства аппаратуры окончания канала данных и оконечного оборудования данных соединяются одним из следующих способов.

Двухточечное соединение – соединение, устанавливаемое между двумя станциями данных для передачи данных (рис. 24).

Рис. 24. Двухточечное соединение

Многоточечное соединение – соединение, устанавливаемое между более чем двумя станциями данных для передачи данных (рис. 25).

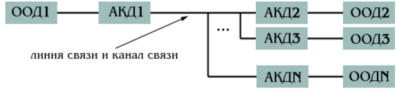

Рис. 25. Многоточечное соединение

2.5. Базовая эталонная модель взаимодействия открытых систем (БЭМ ВОС) (OSI). Открытые информационные сети и их взаимодействие

В основе построения КС лежит жесткая стандартизация. В КС основой стандартизации является многоуровневый подход к разработке средств сетевого взаимодействия. Существует понятие открытые и закрытые информационные системы (сети).

Открытые информационно-компьютерные сети – это информационная сеть, взаимодействия всех входящих в её состав реальных систем, а также, взаимодействие которой с реальными системами, не входящими в ее состав, подчиняются требованиям стандартизации международной организации по стандартизации (МОС) (ISO).

Замечание: Стандарты МОС определяют лишь требования, относящиеся к организации взаимодействия между реальными системами и не ограничивающие их внутреннюю структуру.

В зависимости от сети передачи данных, входящей в состав открытой информационной сети, различают открытые локальные и открытые территориальные информационные сети.

В начале 80-х гг. ряд международных организаций по стандартизации (ISO, ITU-T и некоторые другие) разработали модель, которая сыграла итоговую роль в развитии современных систем. На основе этого многоуровневого подхода и жесткой стандартизации была разработана стандартная семиуровневая модель взаимодействия открытых систем.

2.5.1. Предпосылки создания модели ВОС(OSI). Уровневые протоколы. Сети и модель взаимодействия открытых систем

Уровневые протоколы необходимо применять в современных сетях по следующей причине:

1. Ведущие организации в области сетевых технологий, работающие по их стандартизации, развивают концепции уровневых протоколов.

2. Ведущие фирмы – поставщики сетевых технологий (HP, IBM, DEC, Novell) ориентируются на создание уровневых сетей.

3. Уровневые протоколы имеют хорошее обоснование с точки зрения прикладного применения.

2.5.1.1. Назначение уровневых протоколов

Концепция уровневых протоколов развивалась в течение последних 45 лет и была направлена на решение следующих задач:

1. Обеспечить логическую декомпозицию любой сложной сети на более простые и понятные части (уровни).

2. Обеспечить стандартные интерфейсы между сетевыми функциями.

3. Обеспечить симметрию в отношении функций, реализуемых в каждом узле сети. Каждый уровень в некотором узле сети выполняет те же функции, какие выполняет аналогичный уровень в другом узле.

4. Обеспечить средства предсказания изменений и управления изменениями, которые могут быть внесены в логику работы сети.

5. Обеспечить простой стандартный язык коммуникации разработчиков сети, администраторов, фирм-поставщиков и пользователей.

2.5.1.2. Проблемы проектирования и эксплуатации сетей

В последние годы вычислительные сети усложнились по числу компонент, по логической сложности, по используемому программному обеспечению.

Возникли серьезные проблемы технической эксплуатации сетей.

Проектирование и развитие некоторых сетей происходило без учета каких-либо стандартов, поэтому ее компоненты имели весьма плохо определенные интерфейсы.

У фирм разработчиков был свой подход:

- к проектированию сетей;
- разработке алгоритмов работы сети;
- определению того, каким образом должны встраиваться в сеть интерфейсы конечных пользователей.

Отсутствие унифицированного подхода среди фирм-разработчиков сетевых технологий привело к ситуации несовместимости используемого сетевого оборудования. Необходима была стандартизация сетевых технологий. Следовательно, основная идея разработки общих стандартов для уровневых протоколов заключается в том, чтобы разработать для всех фирм базовые подходы к созданию сетевого КТС и КПО.

2.5.1.3. Описание уровневых протоколов

При описании сетевых протоколов как уровневых используются следующие базовые понятия.

Уровень – компонент иерархической структуры системы (сети), в качестве которого выступает базовая эталонная модель взаимодействия открытых систем и который состоит из подсистем одного ранга.

Услуга уровня – функциональная возможность, которую данный уровень вместе с нижерасположенными (нижележащими) уровнями обеспечивает смежному верхнему уровню.

Сервис уровня – совокупность услуг уровня и правил их использования.

Пользователь сервиса – абстрактное представление совокупности логических объектов уровня в одной открытой системе, которые используют сервис уровня через точку доступа к сервису (ТДС).

Точка доступа к сервису (*ТДС* – SAP) – это точка, в которой логический объект уровня представляет сервис логическому объекту смежного верхнего уровня.

Поставщик сервиса – это абстрактное представление совокупности логических объектов уровня, поставляющих сервис уровня.

Объект – это некоторый специализированный модуль.

Функция – это некоторая подсистема уровня.

Каждая подсистема состоит из логических объектов.

Стандартная технология взаимодействия с уровнем или поставщиком сервиса показана на рис. 26 и 27.

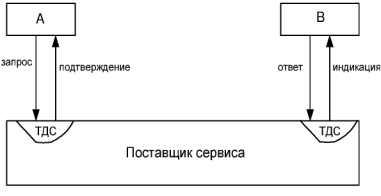

Рис. 26. Стандартная технология сетевого взаимодействия

Посредством ТДС осуществляется вызов в уровень или из уровня четырех транзакций, называемых примитивами:

Запрос – примитив, используемый пользователем сервиса для вызова некоторой функции.

Индикация – примитив, используемый поставщиком сервиса:

- для вызова функций;
- уведомления о том, что функция была вызвана в некоторый ТДС.

Ответ – используется пользователем сервиса для завершения функции, ранее вызванной примитивом индикации в этой ТДС.

Подтверждение – используется поставщиком сервиса для завершения функции, ранее вызванной запросом в этой ТДС.

Рис. 27. Стандартная технология сетевого взаимодействия

В процессе межуровневого взаимодействия используются 5 основных компонент (рис. 28), к которым относятся:

1. **SDU** – сервисный блок данных – это данные пользователя, передаваемые в прозрачном режиме уровнем N+1 в уровень N-1 через уровень N.

2. **PCI** – управляющая информация протокола – это информация, которой обмениваются одноуровневые объекты в различных узлах сети, чтобы сообщить некоторому объекту об необходимости выполнения сервисной функции.

3. **PDU** – протокольный блок данных – это блок данных, определенный протокол взаимодействия одноранговых и логических объектов и содержащий управляющую информацию протокола и данные пользователя.

4. **ICI** – управляющая информация интерфейса – это временной параметр, передаваемый между вышележащим и нижележащим уровнями для вызова сервисных функций.

5. **IDU** – интерфейсный блок данных – это либо полный блок информации или блок данных, передаваемый через TDC некоторого уровня между логическим объектом этого уровня и логическим объектом смежного верхнего уровня, посредством элементарного взаимодействия [1, 22].

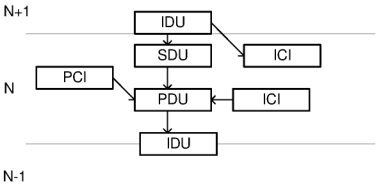

Рис. 28. Компоненты межуровневого взаимодействия

Открытая система – любая система (компьютер, компьютерная сеть, ОС, программный пакет, другие аппаратные и программные продукты), которая построена в соответствии с открытыми спецификациями.

Спецификация (в вычислительной технике) – это формализованное описание аппаратных или программных компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, ограничений и особых характеристик. Не всякая спецификация является стандартом.

Модель ВОС (OSI – Open System Interconnection) определяет различные уровни взаимодействия систем, дает стандартные имена и указывает, какие функции должен выполнять каждый уровень.

Модель взаимодействия открытых систем описывает только системные средства взаимодействия, реализующие ОС, системными и аппаратными средствами.

Архитектура модели ВОС показана на рис. 29 и 30 и включает следующие уровни:

1) физический;
2) канальный;
3) сетевой;
4) транспортный;
5) сеансовый;
6) представительный;
7) прикладной.

Физический уровень – это уровень побитовых протоколов передачи информации. Интерфейс между ЭВМ, участвующими во взаимодействии,

и средой передачи дискретных сигналов управления физическим каналом сводится:

– к выделению начала и конца кадра, несущего в себе передаваемые данные;

– формированию и приему сигналов определенной физической природы со скоростью, соответствующей пропускной способности канала;

– скорости формирования и анализа специальных кодовых последовательностей, характеризующих состояние канала.

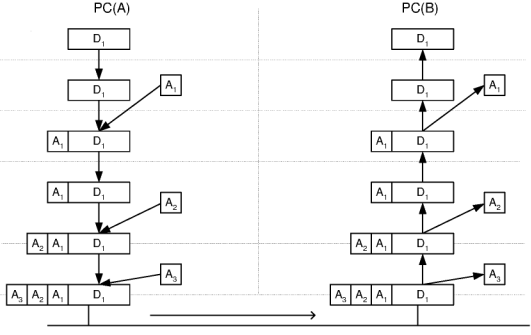

Рис. 29. Архитектура модели ВОС

Стандарты включают рекомендации протоколов X.21 и X.25 МККТ, определяющие следующие характеристики:

- механические;
- электрические;
- функциональные;
- процедурные (необходимы для установления (активизации) поддержания и расторжения (дезактивации) физических соединений).

Физический уровень делится:

- на подуровень передачи физических сигналов PS – выделяется в целях облегчения схемной интеграции с канальным уровнем;

• подуровень интерфейса с устройством доступа AUI, который позволяет размещать PS на некотором расстоянии от носителя, т.е. передающей среды;

• подуровень подключения к физической среде (PMA) – PMA, который согласует сигналы, поступающие из подуровня PS с требованием передающей среды; тем самым обеспечивает возможность использования определенного подуровня PS с несколькими типами передающей среды.

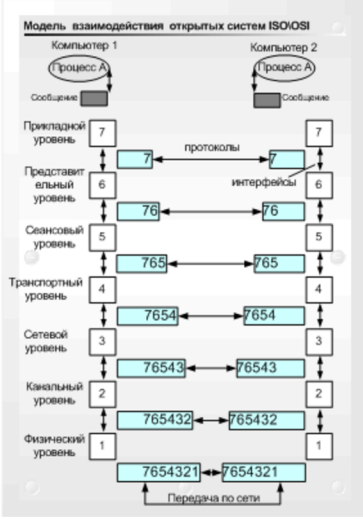

Рис. 30. Архитектура модели ВОС

Канальный уровень – уровень звена данных, который формирует из самых передаваемых физических уровней «информационные кадры» и их последовательности, осуществляет структуризацию передаваемых данных,

управление доступом к передающей среде, обнаруживает и исправляет ошибки.

Канальный уровень обеспечивает:

- синхронизацию данных для разграничения потоков битов из физического уровня;
- вид представления битовых последовательностей, который создает определенные гарантии, чтобы данные благополучно прибыли в ООД;
- управление потоками данных, чтобы гарантировать, что ООД не будет перегружено в любой момент времени слишком большим объемом данных;
- обнаружение ошибок и обеспечение механизма восстановления и их дублирования.

В протоколах канального уровня, используемых в локальных сетях, заложена определенная структура связей между компьютерами и способы их адресации. Хотя канальный уровень и обеспечивает доставку кадра между любыми двумя узлами локальной сети, он это делает только в сети с совершенно определенной топологией связей, именно той топологией, для которой он был разработан. К таким типовым топологиям, поддерживаемым протоколами канального уровня локальных сетей, относятся общая шина, кольцо и звезда, а также структуры, полученные из них с помощью мостов и коммутаторов. Примерами протоколов канального уровня являются протоколы Ethernet, Token Ring, FDDI, lOOVG-AnyLAN.

Сетевой уровень – устанавливает связь в сети между двумя абонентами. Соединение между абонентами происходит в результате выполнения функций маршрутизации. Сообщения сетевого уровня называются пакетами (packets). При организации доставки пакетов на сетевом уровне используется понятие «номер сети». В этом случае адрес получателя состоит из старшей части – номера сети и младшей – номера узла в этой сети. Все узлы одной сети должны иметь одну и ту же старшую часть адреса, поэтому термину «сеть» на сетевом уровне можно дать и другое, более формальное определение: сеть – это совокупность узлов, сетевой адрес которых содержит один и тот же номер сети.

Сетевой уровень обеспечивает:

- обработку ошибок;
- мультиплексирование данных;
- управление потоками данных;

• определяет интерфейс ООД пользования с сетью пакетной коммуникации, а также интерфейс двух взаимодействующих ООД через сеть с пакетной коммуникацией;

• определяет маршрутизацию в сети и межсетевую связь или связь между сетями.

Примером стандарта для уровня является протокол X.25, протокол межсетевого взаимодействия IP стека TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell.

Транспортный уровень. На транспортном уровне осуществляется поддержка между двумя взаимодействующими друг с другом пользовательскими процессами. Транспортный уровень отвечает:

• за качество транспортировки данных;

• корректировку ошибок передачи данных;

• независимость вычислительных сетей;

• сервис транспортировки из конца в конец;

• минимизацию затрат связи, что гарантирует непрерывную безопасность при передаче данных;

• обеспечивает интерфейс между сетью передачи данных и верхними тремя уровнями.

Транспортный уровень проектируется таким образом, чтобы отделить пользователя от некоторых физических и функциональных аспектов пакетной сети.

Протоколы нижних четырех уровней обобщенно называют сетевым транспортом или транспортной подсистемой, так как они полностью решают задачу транспортировки сообщений с заданным уровнем качества в составных сетях с произвольной топологией и различными технологиями. Остальные три верхних уровня решают задачи предоставления прикладных сервисов на основании имеющейся транспортной подсистемы.

В качестве примера транспортных протоколов можно привести протоколы TCP и UDP стека TCP/IP и протокол SPX стека Novell.

Сеансовый уровень. Координирует передачу приема сообщений в течение одного сеанса связи. Служит интерфейсом пользователя с уровнем транспортных услуг. Осуществляет:

• контроль рабочих параметров;

• управление потоками данных промежуточных накопителей и диалоговый контроль;

- гарантированную передачу данных;
- дополнительные функции управления паролями, подсчета платы за использование ресурсами сети;
- точки синхронизации для промежуточного контроля и восстановления при передаче файлов;
- аварийное окончание и рестарты;
- нормальную и ускоренную передачу данных.

Уровень представления данных. Предназначен для интерпретации данных, подготовки данных для пользовательского прикладного уровня. Определяет:

- синтаксис данных или представление данных;
- принимаемые типы данных не прикладного уровня, которыесогласовывались бы с уровнем того же ранга;
- отображение данных на виртуальном терминале;
- услуги приема электронных сообщений из уровня приложения и согласования.

Примером такого протокола является протокол Secure Socket Layer (SSL), который обеспечивает секретный обмен сообщениями для протоколов прикладного уровня стека TCP/IP.

Прикладной уровень. Занимается поддержкой прикладного процесса конечного пользователя; в отличие от уровня представления данных этот уровень определяет семантику данных.

Уровень содержит сервисные элементы для поддержки прикладных процессов, которые используются при обмене финансовыми и деловыми данными. Осуществляет управление ресурсами при административном управлении. То есть это набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые Web-страницы, а также организуют свою совместную работу, например, с помощью протокола электронной почты. Единица данных, которой оперирует прикладной уровень, обычно называется сообщением (message).

Пример служб прикладного уровня в виде файловых служб: NCP в операционной системе Novell NetWare, SMB в Microsoft Windows NT, NFS, FTP и TFTP, входящие в стек TCP/IP.

Технология взаимодействия в рамках модели OSI показана на рис. 29 и рис. 30.

В заключении данного раздела следует отметить особенности различных уровней модели ВОС.

Физический, канальный, сетевой уровни модели ВОС являются сетезависимыми, т.е. протоколы данных уровней тесным образом связаны с технической реализацией ВС и используемым коммуникационным оборудованием. В результате переход на оборудование другого стандарта означает полную смену протоколов физического и канального уровней во всех узлах сети.

Прикладной, представительный, сеансовый ориентированы на приложения и в малой степени зависят от технических особенностей построения ВС. На протоколы этих уровней не влияют: переход на другой сетевой стандарт (сетевую технологию), изменение топологии сети, замена сетевого оборудования.

Транспортный уровень занимает промежуточное положение в архитектуре модели ВОС, он скрывает все детали функционирования нижних уровней от верхних. Это позволяет разрабатывать приложения, которые не зависят от технических средств непосредственной транспортировки сообщений.

На примере модели ВОС показано, что взаимодействие прикладных процессов, или объектов в ВС осуществляется в соответствии с определенными правилами.

Сетевой протокол – это набор синтаксических и семантических правил, в соответствии с которыми осуществляется взаимодействие процессов, или объектов одного уровня.

Межуровневый сетевой интерфейс – это правило взаимодействия процессов, или объектов двух соседних уровней [22].

Контрольные вопросы к разделу 2

1. Какие основные требования предъявляются к ВС при их проектировании?
2. Параметры качества обслуживания в ВС. Что относится к основным параметрам качества обслуживания в ВС?
3. Как сказываются требования к надежности и безопасности на качестве работы сети и конечных пользователей?
4. Что характеризует отказоустойчивость и в какой связи данный показатель находится с надежностью?

5. Что характеризует готовность и непротиворечивость данных ?
6. 6.Что характеризует производительность, пропускная способность, время реакции, задержка передачи?
7. Что характеризует управляемость?
8. Что характеризует расширяемость и масштабируемость?
9. Что характеризует прозрачность и совместимость?
10. Понятие «сетевая архитектура». Какие показатели лежат в основе критерия сравнительной оценки сетевых архитектур?
11. Приведите примеры зарубежных и отечественных стандартов сетевых архитектур.
12. Дайте определение понятию «топология». Что определяет топология ВС?
13. Перечислите основные разновидности топологий, применяемых в ВС.
14. Топология общая шина (магистраль). Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с топологией «общая шина».
15. Топология «звезда». Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с топологией «звезда».
16. Топология «кольцо». Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с кольцевой топологией.
17. Топология иерархическая на базе звезды. Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с данной топологией.
18. Топология «полносвязная». Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с полносвязной топологией.
19. Топология «ячеистая». Достоинства и недостатки. Распространенность данного вида топологии. Примеры ВС с ячеистой топологией.
20. Перечислите базовые составляющие компоненты ВС.
21. Назначение ООД как конечного сетевого оборудования.
22. Назначение АКД или аппаратуры линии связи.
23. В чем различие между АКД и коммуникационным сетевым оборудованием?
24. Приведите классификацию линий связи (ЛС) в зависимости от среды передачи.

25. Охарактеризуйте проводные воздушные ЛС. Достоинства и недостатки.
26. Охарактеризуйте кабельные ЛС.
27. Охарактеризуйте наземные и спутниковые радиоканалы.
28. Перечислите основные составляющие, включающие понятие «коммуникационное сетевое оборудование».
29. Повторитель, функциональное назначение. Основные характеристики, структурно-функциональная схема.
30. Сетевой мост, функциональное назначение. Основные характеристики, структурно-функциональная схема.
31. Коммутатор, функциональное назначение. Основные характеристики, структурно-функциональная схема.
32. Концентратор, функциональное назначение. Основные характеристики, структурно-функциональная схема.
33. Преобразователь интерфейсов, функциональное назначение. Основные характеристики, структурно-функциональная схема.
34. Маршрутизатор, функциональное назначение. Основные характеристики, структурно-функциональная схема.
35. Брандмауэр, функциональное назначение. Основные характеристики, структурно-функциональная схема.
36. Модем, модемный пул, LAN модем, функциональное назначение. Основные характеристики, структурно-функциональная схема.
37. Оконечное оборудование данных (ООД). Состав ООД в ВС.
38. Файловый сервер (ФС). Назначение и особенности применения в ВС. Требования, предъявляемые к ФС.
39. Рабочая станция (РС). Требования, предъявляемые к РС.
40. Типовая структура сети передачи данных (СПД), лежащая в основе построения ВС.
41. Виды соединений в ВС.
42. Каким требованиям должна удовлетворять открытая информационная система?
43. Предпосылки создания модели ВОС (OSI).
44. Описание уровневых протоколов. Понятие «уровень», услуга уровня.
45. Точка доступа к сервису (ТДС), сервис уровня, поставщик сервиса, пользователь сервиса.
46. Перечислите примитивы, используемые при взаимодействии ПП.

47. Перечислите основные информационные компоненты, используемые при межуровневом взаимодействии.
48. Покажите пример взаимодействия ПП с использованием понятия «уровень».
49. Дайте определение открытой системе. Как ВС относятся к открытым системам?
50. Архитектура модели ВОС.
51. Физический уровень модели ВОС. Структура физического уровня. Основные функции, выполняемые на физическом уровне.
52. Канальный уровень модели ВОС. Основные функции, выполняемые на канальном уровне. Пример протоколов канального уровня.
53. Сетевой уровень модели ВОС. Основные функции, выполняемые на сетевом уровне. Пример протоколов сетевого уровня.
54. Транспортный уровень модели ВОС. Основные функции, выполняемые на транспортном уровне. Примеры протоколов транспортного уровня.
55. Сеансовый уровень модели ВОС. Основные функции, выполняемые на сеансовом уровне. Примеры протоколов сеансового уровня.
56. Представительный уровень модели ВОС. Основные функции, выполняемые на представительном уровне.
57. Прикладной уровень модели ВОС. Основные функции, выполняемые на прикладном уровне.
58. Перечислите сетезависимые и сетенезависимые уровни модели. ВОС. Особенности использования сетенезависимых и сетезависимых уровней.
59. Дайте определение понятию «сетевой протокол».
60. Дайте определение понятию «межуровневый сетевой интерфейс».

3. СЕТИ ПЕРЕДАЧИ ДАННЫХ. ОСНОВА ПОСТРОЕНИЯ КС

Даже при рассмотрении простейшей сети, состоящей всего из двух машин, можно увидеть многие проблемы, присущие любой вычислительной сети, в том числе проблемы, связанные с физической передачей сигналов по линиям связи, без решения которой невозможен любой вид связи.

Основой любой ЛС является физическая среда передачи данных (medium), которая может представлять собой кабель, т. е. набор проводов, изоляционных и защитных оболочек и соединительных разъемов, а также земную атмосферу или космическое пространство, через которые распространяются электромагнитные волны.

Как было показано в разд. 2, в зависимости от среды передачи данных линии связи разделяются на проводные (воздушные), кабельные (медные и волоконно-оптические), радиоканалы наземной и спутниковой связи.

Кабельные линии представляют собой достаточно сложную конструкцию. В компьютерных сетях применяются три основных типа кабеля: кабели на основе скрученных пар медных проводов, коаксиальные кабели с медной жилой, а также волоконно-оптические кабели.

Кабельные каналы для целей телекоммуникаций исторически использовались первыми. Да и сегодня по суммарной длине они превосходят даже спутниковые каналы. Основную долю этих каналов, насчитывающих многие сотни тысяч километров, составляют телефонные медные кабели. Эти кабели содержат десятки или даже сотни скрученных пар проводов. Полоса пропускания таких кабелей обычно составляет 3–3,5 кГц при длине 2–10 км. Эта полоса диктовалась ранее нуждами аналогового голосового обмена в рамках коммутируемой телефонной сети. с учетом возрастающих требованиям к широкополосности каналов скрученные пары проводов пытались заменить коаксиальными кабелями, которые имеют полосу от 100 до 500 МГц (до 1 Гбит/с), и даже полыми волноводами. Именно коаксиальные кабели стали в начале транспортной средой локальных сетей ЭВМ (10base-5 и 10base-2).

Следовательно, применительно к кабелям современных ВС различного масштаба необходимо говорить о кабельных системах, сложных коммуникационных компонентах телекоммуникационных систем.

3.1. Основные характеристики ЛС

К основным характеристикам линий связи относятся:

- амплитудно-частотная характеристика;
- полоса пропускания;
- затухание;
- помехоустойчивость;
- перекрестные наводки на ближнем конце линии;
- пропускная способность;
- достоверность передачи данных;
- удельная стоимость.

В первую очередь разработчика вычислительной сети интересуют пропускная способность и достоверность передачи данных, поскольку эти характеристики прямо влияют на производительность и надежность создаваемой сети. Пропускная способность и достоверность – это характеристики как линии связи, так и способа передачи данных. Поэтому если способ передачи (протокол) уже определен, то известны и эти характеристики.

Однако нельзя говорить о пропускной способности линии связи, до того как для нее определен протокол физического уровня. Именно в таких случаях, когда только предстоит определить, какой из множества существующих протоколов можно использовать на данной линии, очень важными являются остальные характеристики линии, такие как полоса пропускания, перекрестные наводки, помехоустойчивость и другие характеристики.

Кроме искажений сигналов, вносимых внутренними физическими параметрами линии связи, существуют и внешние помехи, которые вносят свой вклад в искажение формы сигналов на выходе линии. Несмотря на защитные меры, предпринимаемые разработчиками кабелей и усилительно - коммутирующей аппаратуры, полностью компенсировать влияние внешних помех не удается. Поэтому сигналы на выходе линии связи обычно имеют сложную форму, по которой иногда трудно понять, какая дискретная информация была подана на вход линии.

Степень искажения синусоидальных сигналов линиями связи оценивается с помощью таких характеристик, как амплитудно-частотная характеристика, полоса пропускания и затухание на определенной частоте.

Амплитудно-частотная характеристика (АЧХ) (рис. 31) показывает, как затухает амплитуда синусоиды на выходе линии связи по сравнению с амплитудой на ее входе для всех возможных частот передаваемого сигнала.

Вместо амплитуды в этой характеристике часто используют также такой параметр сигнала, как его мощность.

Знание амплитудно-частотной характеристики реальной линии позволяет определить форму выходного сигнала практически для любого входного сигнала. Для этого необходимо найти спектр входного сигнала, преобразовать амплитуду составляющих его гармоник в соответствии с амплитудно-частотной характеристикой, а затем найти форму выходного сигнала, сложив преобразованные гармоники.

Несмотря на полноту информации, предоставляемой амплитудно-частотной характеристикой о линии связи, ее использование осложняется тем обстоятельством, что получить ее весьма трудно. Ведь для этого нужно провести тестирование линии эталонными синусоидами по всему диапазону частот от нуля до некоторого максимального значения, которое может встретиться во входных сигналах. Причем менять частоту входных синусоид нужно с небольшим шагом, а значит, количество экспериментов должно быть очень большим. Поэтому на практике вместо амплитудно-частотной характеристики применяются другие, упрощенные характеристики – полоса пропускания и затухание.

Рис. 31. Амплитудно-частотная характеристика

Полоса пропускания – это непрерывный диапазон частот, для которого отношение амплитуды выходного сигнала ко входному превышает некоторый заранее заданный предел, обычно равный 0,5. То есть полоса пропускания определяет диапазон частот синусоидального сигнала, при которых этот сигнал передается по линии связи без значительных искажений. Знание полосы пропускания позволяет получить с некоторой степенью приближения тот же результат, что и знание амплитудно-частотной характеристики.

Ширина полосы частот – диапазон частот, в котором может осуществляться передача и обмен данными между абонентами через данный канал связи [9, 10, 22].

Ширина полосы частот эквивалентна понятию ширина полосы пропускания.

Частотный электромагнитный спектр, используемый для передачи данных, имеет следующий вид:

Диапазон частот 10^3 – телефонные речевые частоты (от низких до высоких скоростей передачи);

ДЧ 10^4 – ОНЧ – телефонные речевые частоты (с более высокими скоростями передачи данных);

ДЧ 10^5 – НЧ – коаксиальные подводные кабели (пакетная передача данных с высокой скоростью);

ДЧ 10^6 – СЧ – наземные коаксиальные кабели (звуковая широковещательная передача с амплитудной модуляцией, высокая скорость передачи речи и данных);

ДЧ 10^7 – ВЧ – наземные коаксиальные кабели: коротковолновая широковещательная передача, высокая скорость передачи речи и данных ;

ДЧ 10^8 – ОВЧ – наземные коаксиальные кабели: звуковое и телевизионное вещание с фазовой модуляцией в диапазоне ОВЧ (высокая скорость передачи речи и данных);

ДЧ 10^9 – УВЧ – телевизионное вещание;

ДЧ 10^{10} – СВЧ – короткозамкнутые волноводы: СВЧ-вещание, высокая скорость передачи речи и данных;

ДЧ 10^{11} – КВЧ – спиральные волноводы (высокая скорость передачи данных);

ДЧ 10^{12} – передача в инфракрасном диапазоне (локальная передача данных);

ДЧ 10^{13} – передача в инфракрасном диапазоне (локальная передача данных);

ДЧ 10^{14} – оптоволоконные линии: видимый свет (очень высокая скорость передачи речи и данных);

ДЧ 10^{15} – оптоволоконные линии: ультрафиолетовый диапазон (очень высокая скорость передачи речи и данных);

ДЧ 10^{19} – 10^{23} – рентгеновские и гамма- лучи, данный диапазон для передачи данных не используется.

Затухание (Attenuation**).** Затухание определяется как относительное уменьшение амплитуды или мощности сигнала при передаче по линии сигнала определенной частоты. Таким образом, затухание представляет собой одну точку из амплитудно-частотной характеристики линии связи. Часто при эксплуатации линий заранее известна основная частота передаваемого сигнала, т. е. та частота, гармоника которой имеет наибольшую амплитуду и мощность. Поэтому достаточно знать затухание на этой частоте, чтобы приблизительно оценить искажения передаваемых по линии сигналов. Более точные оценки возможны при знании затухания на нескольких частотах, соответствующих нескольким основным гармоникам передаваемого сигнала.

Таким образом, амплитудно-частотная характеристика, полоса пропускания и затухание являются универсальными характеристиками, и их знание позволяет сделать вывод о том, как через линию связи будут передаваться сигналы любой формы.

Затухание измеряется в децибелах на метр для определенной частоты или диапазона частот сигнала и вычисляется по следующей формуле:

$A = 10 * \lg(P_{вых}/P_{вх})$,

где $P_{вых}$ – мощность сигнала на выходе линии;

$P_{вх}$ – мощность сигнала на входе линии.

Пропускная способность линии. Характеризует максимально возможную скорость передачи данных по линии связи. Пропускная способность измеряется в битах в секунду (бит/с), а также в производных единицах, таких как килобит в секунду (Кбит/с), мегабит в секунду (Мбит/с), гигабит в секунду (Гбит/с) и т. д.

Пропускная способность линии связи зависит не только от ее характеристик, таких как, амплитудно-частотная характеристика, но и от спектра передаваемых сигналов. Если значимые гармоники сигнала (то есть те гармо-

ники, амплитуды которых вносят основной вклад в результирующий сигнал) попадают в полосу пропускания линии, то такой сигнал будет хорошо передаваться данной линией связи и приемник сможет правильно распознать информацию, отправленную по линии передатчиком. Если же значимые гармоники выходят за границы полосы пропускания линии связи, то сигнал будет значительно искажаться, приемник будет ошибаться при распознавании информации, а значит, информация не сможет передаваться с заданной пропускной способностью [9, 10, 22].

Любая линия связи, канал связи имеет ограничения на объем передаваемых данных. Одной из основных причин этого ограничения является *шум*.

Шум в каналах связи – явление, присущее самим каналам, которое не может быть устранено.

Данный шум называется **тепловым, гауссовым, белым** или **фоновым,** происходит от постоянного случайного движения электронов в проводнике.

Шипение, которое слышно в телефонной трубке, является именно таким шумом. Любой проводник является источником шума. Мощность шума пропорциональна полосе пропускания частот (диапазон частот, который позволяет пропустить канал), поэтому расширенная полоса частот содержит увеличенную мощность шумов.

Фундаментальной идеей в *теории связи* является **закон Шеннона**.

Данный закон устанавливает конечные пределы канала передачи данных в виде следующей формулы:

$C=W*\log_2(1+S/N)$,

где C – максимальная пропускная способность (бод);

W – полоса частот;

S/N – отношение мощности сигнала к мощности шумов.

При увеличении полосы частот W, мощности полезного сигнала S, уменьшении уровня шумов N увеличивается пропускная способность каналов передачи данных.

Одним из методов увеличения отношения «сигнал/шум» является размещение большого числа усилителей в канале связи (рис. 32).

Сигнал, проходя по каналу связи, затухает, чтобы этого не наблюдалось, необходимо использовать усилители. Так как уровень шума в канале постоянен, усилители необходимо располагать таким образом и на таком от-

далении друг от друга, чтобы мощность сигнала не упала ниже определенного уровня.

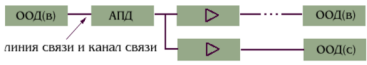

Рис. 32. Подключение усилителей в сеть

Близкое расположение усилителей улучшает соотношение S/N , но это дорогостоящее мероприятие, а сами усилители должны быть тщательно спроектированы для минимизации шума при усилении сигнала.

Другой способ борьбы с тепловым шумом, а также с другими видами шумов, например разрядными видами помех, флуктуациями мощности, приемники в системах связи должны проверять данные, и в случаях обнаружения нарушений запрашивать повторную передачу.

Нарушения (*ошибки*) можно широко классифицировать как *случайные, импульсные* или *смешанные*.

В каналах со случайными ошибками для каждого бита передаваемых данных существует вероятность P неправильного приема и вероятность (1 – P) правильного приема.

Каналы с импульсными ошибками демонстрируют состояние, свободное от ошибок большую часть времени, но иногда появляются групповые или разовые ошибки. Объектом таких ошибок являются радиосигналы, так же, как кабели и провода, например телефонные каналы из витых проводных пар.

Перекрестные наводки. Обычно рассматриваются два случая перекрестных наводок:

1) источник сигнала и приемник находятся по одну сторону кабеля (NEXT – near end crosstalk);

2) приемник и источник находятся на разных концах кабеля (FRXT – far end crosstalk).

NEXT-наводки при большом числе пар проводов в кабеле подчиняются закону $f^{1.5}$, а их уровень составляет около 55 дБ при частоте 100 кГц. FEXT-наводки сильно зависят от схемы коммутации и разводки проводов и обычно менее опасны, чем NEXT. Еще одним источником наводок является импульсный шум внешних электромагнитных переходных процессов. Этот вид наводок обычно характеризуется промежутком времени, в тече-

ние которого его уровень превышает порог чувствительности, и варьируется в зависимости от обстоятельств в очень широких пределах.

NEXT измеряются в децибелах для определенной частоты сигнала.

Импеданс (*волновое сопротивление*) – это полное (активное и реактивное) сопротивление в электрической цепи. Импеданс измеряется в Омах и является относительно постоянной величиной для кабельных систем (например, для коаксиальных кабелей, используемых в стандартах Ethernet, импеданс кабеля должен составлять 50 Ом). Для неэкранированной витой пары наиболее часто используемые значения импеданса – 100 и 120 Ом. В области высоких частот (100–200 МГц) импеданс зависит от частоты.

Активное сопротивление – это сопротивление постоянному току в электрической цепи. В отличие от импеданса активное сопротивление не зависит от частоты, но зависит от длины кабельной системы.

Емкость – это свойство металлических проводников накапливать энергию. Большая величина емкости в кабельной системе приводит к тому, что передаваемые сигналы искажаются, а полоса пропускания линии связи ограничивается или сужается.

Диаметр или площадь сечения проводников, используемых в кабельных системах. Основной материал – медь. Для медных проводников применяется американская система обозначения проводников AWG. Чем больше номер типа проводника, тем меньше его физический диаметр. В Международном и Европейском стандартах диаметр проводников канала связи обозначается в миллиметрах (мм).

3.1.1. Уровень внешнего электромагнитного излучения или электромагнитный шум

Электромагнитный шум:

- фоновый;
- импульсный;
- низкочастотный;
- среднечастотный;
- высокочастотный.

Источники фона:

- источниками фонового электромагнитного шума в диапазоне до 150 кГц могут быть линии электропередачи, телефонные аппараты и лампы дневного света;

• от 150 кГц до 20 Мгц – компьютеры, принтеры, ксероксы;

• от 20 МГц до 1 ГГц – радио- и телевизионные передатчики и микроволновые печи. Основным источником электромагнитного шума являются: моторы, переключатели и сварочные агрегаты [9, 10, 22].

3.2. Кабельные системы

Существует большое разнообразие кабелей для вычислительных сетей, отличающихся по внешнему виду, назначению, характеристикам и цене.

Наиболее популярные системы организации кабельных соединений в вычислительных сетях (локальных вычислительных сетях) предложены фирмами IBM, AT&T, DEC.

Первоначально в качестве среды передачи данных использовались телефонные линии связи, это касается сетей передачи данных. С появлением средств ВТ телефонные линии связи стали использоваться для обмена данными между ЭВМ в вычислительных сетях. Следует отметить, что и в настоящее время телефонные линии связи широко используются в вычислительных сетях различного ранга в зависимости от территориальной распределенности.

3.2.1. Стандарты кабельных систем

В компьютерных сетях применяются кабели, удовлетворяющие определенным стандартам, что позволяет строить кабельную систему сети из кабелей и соединительных устройств разных производителей. Сегодня наиболее употребительными стандартами в мировой практике являются следующие.

Американский стандарт **EIA/TIA-568A**, который был разработан совместными усилиями нескольких организаций: ANSI, EIA/TIA и лабораторией Underwriters Labs (UL).

Международный стандарт **ISO/IEC 11801.**

Европейский стандарт **EN-50173.**

Эти стандарты близки между собой и по многим позициям предъявляют к кабелям идентичные требования. Однако есть и различия между этими стандартами, например, в международный стандарт 11801 и европейский EN-50173 вошли некоторые типы кабелей, которые отсутствуют в стандарте EIA/TAI-568A.

До появления стандарта EIA/TIA большую роль играл американский стандарт системы категорий кабелей Underwriters Labs, разработанный совместно с компанией Anixter. Позже этот стандарт вошел в стандарт EIA/TIA-568.

Кроме этих открытых стандартов, многие компании в свое время разработали свои фирменные стандарты, из которых до сих пор имеет практическое значение только один – стандарт компании IBM.

При стандартизации кабелей принят протокольно-независимый подход. Это означает, что в стандарте оговариваются электрические, оптические и механические характеристики, которым должен удовлетворять тот или иной тип кабеля или соединительного изделия – разъема, кроссовой коробки и т.п. Однако для какого протокола предназначен данный кабель, стандарт не оговаривает. Поэтому нельзя приобрести кабель для протокола Ethernet или FDDI, нужно просто знать, какие типы стандартных кабелей поддерживают протоколы Ethernet и FDDI [9, 10, 22].

3.2.1.1. Кабельные системы компьютерных сетей на основе коаксиального кабеля

Коаксиальный кабель – тип электрического кабеля, в котором центральный провод, окруженный изоляцией, окружен металлическим плетеным экраном. Оси центрального провода и экрана совпадают, что объясняет термин коаксиальный. Коаксиальные кабели обладают широкой полосой пропускания и могут передавать большие объемы данных, речевых и видеосигналов одновременно.

Коаксиальный кабель представляет собой кабель, имеющий центральный провод, или жилу, и оболочку, которые разделены изолирующим слоем (рис. 33).

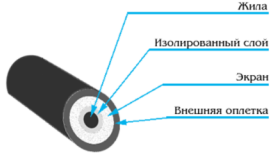

Рис. 33. Структура коаксиального кабеля

Коаксиальные кабели находят широчайшее применение в бытовой радио- и телевизионной технике. Коаксиальный кабель имеет среднюю цену, хорошую помехозащищенность и может применяться для связи на больших расстояниях (несколько км). Скорость передачи данных от 1 до 10 Мбит/с.

В настоящее время имеется сетевое оборудование, позволяющее обеспечить передачу данных по коаксиальному кабелю со скоростями до 50–100 Мбит/с. При использовании широкополосного коаксиального кабеля можно обеспечить скорость передачи данных в сети 500 Мбит/с (500 Мбод). Коаксиальный кабель – это тип электрокабеля, в котором центральный провод (жила), защищенный изоляцией, окружен металлическим (плетеным) экраном.

Существует большое количество типов коаксиальных кабелей, используемых в сетях различного типа – телефонных, телевизионных и компьютерных. Ниже приводятся основные типы и характеристики этих кабелей.

1) **RG-8 и RG-11** – "толстый" коаксиальный кабель, разработанный для сетей Ethernet 10 Base-5. Имеет волновое сопротивление 50 Ом и внешний диаметр 0,5 дюйма (около 12 мм). Этот кабель имеет достаточно толстый внутренний проводник диаметром 2,17 мм, который обеспечивает хорошие механические и электрические характеристики (затухание на частоте 10 МГц). Этот кабель сложно монтировать – он плохо гнется.

2) **RG-58/U, RG-58 A/U и RG-58 C/U** – разновидности "тонкого" коаксиального кабеля для сетей Ethernet 10 Base-2. Кабель RG-58/U имеет сплошной внутренний проводник, а кабель RG-58 A/U – многожильный. Кабель RG-58 C/U проходит "военную приемку". Все эти разновидности кабеля имеют волновое сопротивление 50 Ом, но обладают худшими механическими и электрическими характеристиками по сравнению с "толстым" коаксиальным кабелем. Тонкий внутренний проводник 0,89 мм не так прочен, зато обладает гораздо большей гибкостью, удобной при монтаже. Затухание в этом типе кабеля выше, чем в "толстом" коаксиальном кабеле, что приводит к необходимости уменьшать длину кабеля для получения одинакового затухания в сегменте. Для соединения кабелей с оборудованием используется разъем типа BNC.

3) **RG-59** – телевизионный кабель с волновым сопротивлением 75 Ом. Широко применяется в кабельном телевидении.

4) **RG-62** – кабель с волновым сопротивлением 93 Ома, использовался в сетях Arcnet, оборудование которых сегодня практически не выпускается.

Коаксиальные кабели с волновым сопротивлением 50 Ом (то есть "тонкий" и "толстый") описаны в стандарте EIA/TIA-568. Новый стандарт EIA/TIA-568A коаксиальные кабели не описывает, как морально устаревшие.

Основные характеристики коаксиальных кабелей

- Коаксиальные кабели различных стандартов находят широкое применение в бытовой, радио- и телевизионной технике, в ВС и СПД.
- Коаксиальный кабель имеет среднюю цену и может применяться для связи на большие расстояния (несколько км).
- Обладает хорошей помехоустойчивостью.
- Наиболее распространенный диапазон скоростей передачи данных составляет 10÷100 Мбод.
- При использовании широкополосных коаксиальных кабелей скорость передачи данных может достигать 100÷500 Мбод.

Достоинства коаксиального кабеля:

- широкая полоса пропускания;
- невысокая цена;
- хорошая помехозащищенность;
- высокая скорость передачи данных;
- удобство и технологичность в эксплуатации.

Система DEC Connect фирмы DEC

Концепция данной системы базируется на применении тонкого 50-омного кабеля и вобрала в себя многие лучшие достижения, используемые в системах типа VAX [9, 10, 22].

3.2.1.2. Кабельные системы компьютерных сетей на основе витой пары

Витая пара – кабель, в котором изолированная пара проводников скручена с небольшим числом витков на единицу длины.

Витая пара проводников выполняется с целью уменьшения электрических помех извне при распространении сигналов по кабелю. Витая пара может быть как экранированной (shielded), так и неэкранированной (unshielded), вид кабелей приведен на рис. 3.4. Терминология конструкций экрана неоднозначна, здесь используются слова braid (оплетка), shield и screen (экран, защита), foil (фольга), tinned drain wire (луженый «дренажный» провод, идущий вдоль фольги и слегка ее обвивающий).

Кабельные системы компьютерных сетей на основе неэкранированной витой пары (UTP)

Неэкранированная витая пара (НВП) больше известна по аббревиатуре UTP (Unshielded Twisted Pair). Если кабель заключен в общий экран, но пары не имеют индивидуальных экранов, то, согласно стандарту (ISO 11801), он тоже относится к неэкранированным витым парам и обозначается UTP или SUTP. К ним же относится ScTP (Screened Twisted Pair) или FTP (Foiled Twisted Pair) – кабель, в котором витые пары заключены в общий экран из фольги, а также SFTP – кабель, у которого общий экран состоит из фольги и оплетки.

Медный неэкранированный кабель UTP в зависимости от электрических и механических характеристик разделяется на 5 категорий. Кабели категорий 1 и 2 были определены в стандарте EIA/TIA-568, но в стандарт 568A уже не вошли, как устаревшие.

Кабели категории 1 (Category1) применяются там, где требования к скорости передачи минимальны. Обычно это кабель для цифровой и аналоговой передачи голоса и низкоскоростной (до 20 Кбит/с) передачи данных. До 1983 г. это был основной тип кабеля для телефонной разводки.

Кабели категории 2 были впервые применены фирмой IBM при построении собственной кабельной системы. Главное требование к кабелям этой категории – способность передавать сигналы со спектром до 1 МГц.

Кабели категории 3 были стандартизованы в 1991 г. когда был разработан Стандарт телекоммуникационных кабельных систем для коммерческих зданий (EIA-568), на основе которого затем был создан действующий стандарт EIA-568A. Стандарт EIA-568 определил электрические характеристики кабелей категории 3 для частот в диапазоне до 16 МГц, поддерживающих, таким образом, высокоскоростные сетевые приложения. Кабель категории 3 предназначен как для передачи данных, так и для передачи голоса. Шаг скрутки проводов равен примерно 3 витка на 1 фут. Кабели категории 3 сейчас составляют основу многих кабельных систем зданий, в которых они используются для передачи и голоса, и данных.

Кабели категории 4 представляют собой несколько улучшенный вариант кабелей категории 3. Они обязаны выдерживать тесты на частоте передачи сигнала 20 МГц и обеспечивать повышенную помехоустойчивость и низкие потери сигнала. Кабели категории 4 хорошо подходят для применения в системах с увеличенными расстояниями (до 135 метров) и в сетях

Token Ring с пропускной способностью 16 Мбит/с. На практике используются редко.

Кабели категории 5 были специально разработаны для поддержки высокоскоростных протоколов. Их характеристики определяются в диапазоне до 100 МГц. Большинство новых высокоскоростных стандартов ориентируются на использование витой пары 5 категории. На этом кабеле работают протоколы со скоростью передачи данных 100 Мбит/с – FDDI (с физическим стандартом TP-PMD), Fast Ethernet, 100VG-AnyLAN, а также более скоростные протоколы – ATM на скорости 155 Мбит/с и Gigabit Ethernet на скорости 1000 Мбит/с (вариант Gigabit Ethernet на витой паре категории 5 стал стандартом в июне 1999 г.). Кабель категории 5 пришел на замену кабелю категории 3, и сегодня все новые кабельные системы крупных зданий строятся именно на этом типе кабеля совместно с оптоволоконными системами.

Наиболее важные электромагнитные характеристики кабеля категории 5 имеют следующие значения:

- полное волновое сопротивление в диапазоне частот до 100 МГц равно 100 Ом (стандарт ISO 11801 допускает также кабель с волновым сопротивлением 120 Ом);
- величина перекрестных наводок NEXT в зависимости от частоты сигнала должна принимать значения не менее 74 Дб на частоте 150 кГц и не менее 32 Дб на частоте 100 МГц;
- затухание имеет предельные значения от 0,8 Дб (на частоте 64 кГц) до 22 Дб (на частоте 100 МГц);
- активное сопротивление не должно превышать 9,4 Ом на 100 м;
- емкость кабеля не должна превышать 5,6 нФ на 100 м.

Все кабели UTP независимо от их категории выпускаются в четырехпарном исполнении. Каждая из четырех пар кабеля имеет определенный цвет и шаг скрутки. Обычно две пары предназначены для передачи данных, а две – для передачи голоса [9, 10, 22].

Кабельные системы компьютерных сетей на основе экранированной витой пары (STP).

Система стандартов кабельных систем была впервые опубликована в 1984 г. и была использована в качестве основы для ЛВС "Token Ring". Это стандарт сети, выпускаемый IBM. Экранированная витая пара (ЭВП), она

же STP (Shielded Twisted Pair), имеет множество разновидностей, но каждая пара обязательно имеет собственный экран:

STP с обозначением вида «*Type xx*» – «классическая» витая пара, введенная IBM для сетей Token Ring. Каждая пара этого кабеля заключена в отдельный экран из фольги (кроме типа 6A), обе пары заключены в общий плетеный проволочный экран, снаружи всё покрыто изоляционным чулком, импеданс – 150 Ом. Провод может быть одножильным или многожильным калибра 22–26 AWG. Одножильный кабель 22 AWG может иметь полосу пропускания до 300 МГц.

STP категории 5 – общее название для кабеля с импедансом 100 Ом, имеющего отдельный экран для каждой пары и различное исполнение (фольга, оплетка, их комбинация). Иногда под этим же названием идет кабель, имеющий только общий экран (фирма AMP).

PiMF (Pair in Metal Foil) – кабель, в котором каждая пара завернута в полоску металлической фольги, а все пары находятся в общем экранирующем чулке. От STP Type 1 этот кабель отличается числом пар (здесь их четыре) и более широкой полосой частот (выпускается и на 600 МГц).

SSTP (Shielded-Screened Twisted Pair) категории 7 – кабель, аналогичный PiMF.

Наибольшее распространение получили кабели с числом пар 2 и 4. Существуют и двойные конструкции — два кабеля по две или четыре пары заключены в смежные изоляционные чулки. В общий чулок могут быть заключены и кабели STP+UTP. Из многопарных популярны 25-парные, а также сборки по 6 штук 4-парных. Кабели с большим числом пар (50, 100) применяются только в телефонии, поскольку изготовление многопарных кабелей высоких категорий – задача очень сложная.

Каждая пара кабеля имеет свой шаг скрутки, отличающийся от соседних. Этим обеспечивается снижение взаимной индуктивности и емкости проводов пар, а следовательно, и снижение перекрестных наводок. Поскольку от шага скрутки зависят волновые характеристики пары, пары в кабеле не идентичны. Каждая пара в отрезке кабеля имеет свою «электрическую длину», определяемую через время распространения сигнала и номинальную (для данного кабеля) скорость распространения волны. Иногда применяют переменный шаг скрутки для каждой пары — это выравнивает усредненные параметры пар при сохранении допустимого уровня перекрестных помех.

Кабели чаще всего бывают круглыми – в них элементы собираются в пучок. Существуют и плоские кабели, используемые в телефонии для подключения оконечного оборудования, но в них пары проводов обычно не скручены, так что высокие рабочие частоты для них не реализуемы. Существуют и плоские специальные кабели для прокладки коммуникаций под ковровыми покрытиями, среди которых есть и кабели категорий 3 и 5.

Проводники могут быть жесткими одножильными (solid) или гибкими многожильными (stranded или flex), состоящими обычно из 7 проволочек (7-strand). Кабель с одножильными проводами обладает лучшими и более стабильными характеристиками. Его применяют в основном для стационарной проводки, которая составляет наибольшую часть в кабельных линиях. Гибкий многожильный кабель применяют для соединения оборудования со стационарной проводкой и коммутационных шнуров [9, 10, 22].

Стандарт IBM

Основным стандартом, определяющим параметры экранированной витой пары, является фирменный стандарт IBM. В этом стандарте кабели делятся на типы: Type 1, Type 2,..., Type 9.

Основным типом экранированного кабеля является кабель Type 1 стандарта IBM. Он состоит из 2-х пар скрученных проводов, экранированных проводящей оплеткой, которая заземляется. Электрические параметры кабеля Type 1 примерно соответствуют параметрам кабеля UTP категории 5. Однако волновое сопротивление кабеля Type 1 равно 150 Ом (UTP категории 5 имеет волновое сопротивление 100 Ом), поэтому простое «улучшение» кабельной проводки сети путем замены неэкранированной пары UTP на STP Type 1 невозможно. Трансиверы, рассчитанные на работу с кабелем, имеющим волновое сопротивление 100 Ом, будут плохо работать на волновое сопротивление 150 Ом. Поэтому при использовании STP Type 1 необходимы соответствующие трансиверы. Такие трансиверы имеются в сетевых адаптерах Token Ring, так как эти сети разрабатывались для работы на экранированной витой паре. Некоторые другие стандарты также поддерживают кабель STP Type 1 (например, lOOVG-AnyLAN), а также Fast Ethernet. В случае, если технология может использовать UTP и STP, нужно убедиться, на какой тип кабеля рассчитаны приобретаемые трансиверы. Сегодня кабель STP Type 1 включен в стандарты EIA/TIA-568A, ISO 11801 и EN50173, т. е. приобрел международный статус.

Экранированные витые пары используются также в кабеле IBM Type 2, который представляет кабель Type 1 с добавленными двумя парами неэкранированного провода для передачи голоса.

Для присоединения экранированных кабелей к оборудованию используются разъемы конструкции IBM.

Здесь предусмотрено использование следующих кабельных систем.

Кабель типа 1. Используется для передачи данных. Это кабель с медной основой. Состоит из двух витых пар, изготовлен из жестких проводников и размещенных во втором экране из фольги. Сверху покрыт изоляционным слоем. Кабели данного типа используются для соединений терминалов, расположенных в производственных помещениях.

Кабель типа 2. Предназначен для передачи данных и телефонной связи. Он аналогичен предыдущему, но имеет еще четыре витых пары. Конструктивно выпускается в обычном и особо защищенном исполнении.

Кабель типа 3. Этот телефонный кабель состоит из четырех витых пар с внешней изоляцией. Так как данный тип кабеля не защищен, то он более подвержен помехам.

Кабель типа 5. Оптоволокно. Данный тип состоит из 2-х волокон 100 и 140 микрон. 100-микронное волокно – сердцевина, которая окружена 140-микронной оболочкой. Сердцевина и оболочка – многомодовые (не относится к витой паре).

Кабель типа 6. Монтажный кабель. Данный тип кабеля предназначен для соединения с розетками. Состоит из 2-х витых пар на основе стандартного проводника и является более гибким и удобным.

Кабель типа 8. Нужен для прокладки коммуникаций под коврами. Удобен для применения в офисах или рабочих помещениях, где нет капитальных стен. Состоит из 2-х витых пар с жесткими проводниками в плоской оболочке.

Кабель типа 9. Более дешевая версия кабеля типа 1 с уменьшенными на 1/3 предельными расстояниями обмена данными. Состоит из двух витых пар и гибких проводников.

Система PDS AT&T

Система базируется на неэкранированных витых парах, применяемых в телефонии. Передает речь/данные. Компоненты данной системы являются более дешевыми по сравнению со стандартом IBM.

Особое место занимают кабели категорий 6 и 7, которые промышленность начала выпускать сравнительно недавно. Для кабеля категории 6 характеристики определяются до частоты 200 МГц, а для кабелей категории 7 – до 600 МГц. Кабели категории 7 обязательно экранируются, причем как каждая пара, так и весь кабель в целом. Специалисты в области построения и проектирования сетей сомневаются в необходимости применения кабелей категории 7. По стоимости выходит на уровень волоконно-оптических кабелей, а по технологическим показателям ниже.

Витая пара имеет следующие технические характеристики:

- скорость передачи до 100 Мбит/с. В настоящее время внедряются разработки, у которых скорость передачи до 1000 Мбод;
- длина кабеля может быть больше 1 км при скорости передачи данных 1 Мбод;
- низкая цена;
- простота монтажа и установки;
- высокая помехоустойчивость и защищенность при использовании экранированной витой пары;
- относительная простота при обслуживании.

Пример конструктивного исполнения витых пар показан на рис. 34.

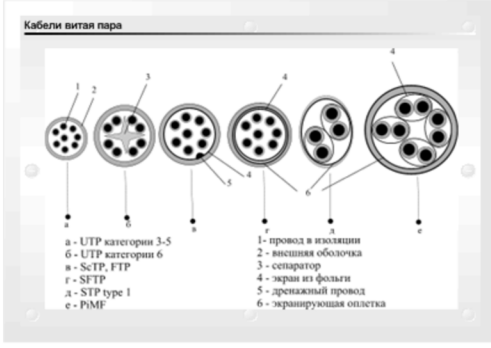

Рис. 34. Конструктивное исполнение витых пар

Наибольшее распространение получили кабели с числом пар 2 и 4. Существуют и двойные конструкции – два кабеля по две или четыре пары заключены в смежные изоляционные чулки. В общий чулок могут быть заключены и кабели STP+UTP. Из многопарных популярны 25-парные, а также сборки по 6 штук 4-парных. Кабели с большим числом пар (50, 100) применяются только в телефонии, поскольку изготовление многопарных кабелей высоких категорий – задача очень сложная.

Каждая пара кабеля имеет свой шаг скрутки, отличающийся от соседних. Этим обеспечивается снижение взаимной индуктивности и емкости проводов пар, а следовательно, и снижение перекрестных наводок. Поскольку от шага скрутки зависят волновые характеристики пары, пары в кабеле не идентичны. Каждая пара в отрезке кабеля имеет свою «электрическую длину», определяемую через время распространения сигнала и номинальную (для данного кабеля) скорость распространения волны. Иногда применяют переменный шаг скрутки для каждой пары – это выравнивает усредненные параметры пар при сохранении допустимого уровня перекрестных помех.

Кабели чаще всего бывают круглыми – в них элементы собираются в пучок. Существуют и плоские кабели, используемые в телефонии для подключения оконечного оборудования, но в них пары проводов обычно не скручены, так что высокие рабочие частоты для них не реализуемы. Существуют и плоские специальные кабели для прокладки коммуникаций под ковровыми покрытиями, среди которых есть и кабели категорий 3 и 5.

Проводники могут быть жесткими одножильными (solid) или гибкими многожильными (stranded или flex), состоящими обычно из 7 проволочек (7-strand). Кабель с одножильными проводами обладает лучшими и более стабильными характеристиками. Его применяют в основном для стационарной проводки, которая составляет наибольшую часть в кабельных линиях. Гибкий многожильный кабель применяют для соединения оборудования со стационарной проводкой и коммутационных шнуров.

Соединительная аппаратура

Соединительная аппаратура обеспечивает возможность подключения к кабелям, то есть предоставляет кабельные интерфейсы. Для витой пары имеется широкий ассортимент коннекторов, предназначенных как для неразъемного, так и разъемного соединения проводов, кабелей и шнуров. Из неразъемных коннекторов распространены соединители типов S110, S66 и Krone, являющиеся промышленными стандартами. Среди разъемных

наиболее популярны стандартизованные модульные соединители (RJ-11, RJ-45 и др.). Встречаются и коннекторы фирмы IBM, введенные с сетями Token Ring, а также некоторые специфические нестандартизованные коннекторы. Многопарные кабели часто соединяют 25-парными разъемами Telco (RJ-21). Для соединения кабелей и проводов в муфтах применяют различные сплайсы. Большинство коннекторов устанавливаются без применения пайки и по принципу обеспечения контакта относятся к классу IDC.

Сплайсы предназначены для быстрого сращивания или разветвления кабелей в тех местах, где впоследствии не потребуется перекоммутация. В основном они применяются в телефонной части проводки. Вместо пайки, традиционной для телефонии, в них используется контакт со смещением изоляции (IDC). Эти соединители выпускаются как в сухом исполнении, так и с заполнением водоотталкивающим гелем.

Разветвители и адаптеры — это пассивные устройства, устанавливаемые между розетками стационарной кабельной системы и интерфейсами оборудования. В их функции входит «преобразование» типа разъема, а иногда и типа соединяемых кабелей. Конструктивное исполнение может быть жестким или мягким, мягкое исполнение удобнее [5,6, 13, 22].

3.2.1.3. Волоконно-оптические линии связи (ВОЛС). Кабельные системы на базе ВОЛС

Волоконно-оптические кабели состоят из центрального проводника света (сердцевины) – световода в виде стеклянного волокна, окруженного другим слоем стекла – оболочкой, обладающей меньшим показателем преломления, чем сердцевина. Распространяясь по сердцевине, лучи света не выходят за ее пределы, отражаясь от покрывающего слоя оболочки.

Световод (сердцевина в оболочке) с защитным покрытием называется оптическим волокном. Оптоволокно в первую очередь характеризуется диаметрами сердцевины и оболочки, эти размеры в микрометрах записываются через дробь: 50/125, 62,5/125, 100/140, 8/125, 9,5/125 мкм. Наружный диаметр волокна (с покрытием) тоже стандартизован, в телекоммуникациях в основном используются волокна с диаметром 250 мкм. Применяются также и волокна с буферным покрытием, или просто буфером (buffer), диаметром 900 мкм, нанесенным на первичное 250-мкм покрытие.

В зависимости от траекторий распространения света различают одномодовое и многомодовое волокно.

Многомодовое волокно (multi mode fiber, MMF) имеет довольно большой диаметр сердцевины – 50 или 62,5 мкм при диаметре оболочки 125 или 100 мкм при оболочке 140 мкм.

Одномодовое волокно (single mode fiber, SMF) имеет диаметр сердцевины 8 или 9,5 мкм при том же диаметре оболочки. Снаружи оболочка имеет защитное покрытие (coating) толщиной 60 мкм, называемое также защитной оболочкой.

Для одномодовых кабелей применяются только полупроводниковые лазеры, так как при таком малом диаметре оптического волокна световой поток, создаваемый светодиодом, невозможно без больших потерь направить в волокно. Для многомодовых кабелей используются более дешевые светодиодные излучатели. Для передачи информации применяется свет с длиной волны 1 550 (1,55), 1300 (1,3) и 850 нм (0,85 мкм). Светодиоды могут излучать свет с длиной волны 850 и 1300 нм. Излучатели с длиной волны 850 нм существенно дешевле, чем излучатели с длиной волны 1300 нм, но полоса пропускания кабеля для волн 850 нм уже, например 200 МГц/км вместо 500 МГц/км.

Лазерные излучатели работают на длинах волн 1 300 и 1 550 нм. Быстродействие современных лазеров позволяет модулировать световой поток с частотами 10 ГГц и выше. Лазерные излучатели создают когерентный поток света, за счет чего потери в оптических волокнах становятся меньше, чем при использовании некогерентного потока светодиодов.

Использование только нескольких длин волн для передачи информации в оптических волокнах связано с особенностью их амплитудно-частотной характеристики. Именно для этих дискретных длин волн наблюдаются ярко выраженные максимумы передачи мощности сигнала, а для других волн затухание в волокнах существенно выше.

Световой импульс, проходя по волокну, из-за явления дисперсии изменит свою форму – «размажется». Дисперсия бывает трех видов: модовая, молекулярная и волноводная.

Модовая дисперсия (modal dispersion) в многомодовом волокне возникает из-за разности длин путей, проходимых лучами различных мод. Эта дисперсия определяется как разность времени прохождения единицы длины волокна различными модами. Ее можно уменьшать, сокращая количество мод, т.е. уменьшая диаметр сердцевины.

Спектральная дисперсия, называемая также молекулярной или материальной, вызвана тем, что волны с разной длиной распространяются в одной и той же среде с различной скоростью, что обусловлено особенностями молекулярной структуры. Поскольку источник излучает не одну волну, а спектр (пусть и узкий), лучи различной длины волны будут достигать приемника не одновременно. Молекулярная дисперсия определяется как разность времени прохождения по волокну излучения различных длин волн, отнесенная к разности длин этих волн и длине волокна. Снизить ее влияние можно уменьшением ширины полосы излучения источника и выбором оптимальной длины волны.

Волноводная дисперсия, актуальная для одномодового волокна, обусловлена разностью скоростей распространения волн по сердцевине и оболочке.

Режим передачи – одномодовый или многомодовый – определяется способом ввода света в волокно, конструкцией волокна и длиной волны источника. Ввод света для одномодового режима должен осуществляться узким лучом точно вдоль оси волокна, здесь в качестве источника можно использовать только лазер. Для многомодовой передачи может использоваться и более дешевый светодиодный излучатель, имеющий более широкую диаграмму направленности. Передача в одномодовом режиме возможна лишь при длине волны, превышающей некоторое пороговое значение. Эта пороговая длина волны определяется конструкцией волокна (диаметром сердцевины). При одномодовой передаче луч передается и по внутренней части оболочки, поэтому ее прозрачность, как и прозрачность сердцевины, влияет на затухание сигнала. Здесь световой луч характеризуется диаметром модового пятна – области сечения волокна, через которую он распространяется. В многомодовом волокне через оболочку свет не идет, так что ее прозрачность несущественна.

Для многомодового волокна существует понятие равновесного распределения мод – РРМ, ему соответствует английский термин EMD (equilibrium mode distribution). Эффективность переноса энергии в разных модах различна – потери в высоких модах больше потерь в низких. В реальных волокнах из-за изгибов и неоднородностей по мере движения свет может переходить из одной моды в другую. В переполненном волокне в переносе энергии участвуют и неэффективные моды. В *ненаполненном* волокне используются только моды низких порядков. Изначально модовое

распределение определяется источником света: светодиод обычно переполняет волокно, лазер не наполняет волокно.

Многомодовое волокно делится на волокно со ступенчатым изменением показателя преломления и волокно с плавным изменением показателя преломления.

Многомодовое волокно со ступенчатым изменением показателя преломления показано на рис. 35.

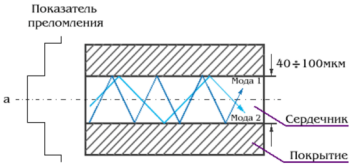

Рис. 35. Структура оптоволокна

Многомодовое волокно с плавным изменением показателя преломления показано на рис. 36.

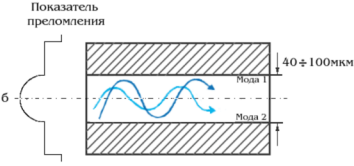

Рис. 36. Структура оптоволокна

Одномодовое волокно показано на рис. 37.

Кабели, применяемые в сетях, используют одномодовые и многомодовые волокна с номинальным диаметром оболочки 125 мкм в покрытии с наружным диаметром 250 мкм, которые могут быть заключены и в 900-мкм буфер. Оптический кабель состоит из одного или нескольких волокон, буферной оболочки, силовых элементов и внешней оболочки. В зависимости

от внешних воздействий, которым должен противостоять кабель, эти элементы выполняются по разному.

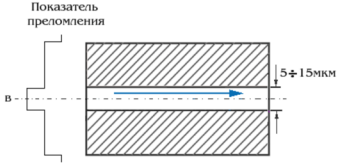

Рис. 37. Структура оптоволокна

По количеству волокон кабели подразделяют на *симплексные* (одножильные), *дуплексные* (2 волокна) и *многожильные* (от 4 до нескольких сотен волокон). В многожильных кабелях обычно применяются однотипные волокна, хотя производители кабеля под заказ могут комплектовать его и разнотипными (MM и SM) волокнами [11,23].

Буфер отделяет волокно от остальных элементов кабеля и является первой ступенью защиты волокна. Буфер может быть плотным или пустотелым. *Плотный буфер* (tight buffer) заполняет все пространство между покрытием и внешней оболочкой кабеля. Простейшим плотным буфером является 900-мкм защитное покрытие волокна. Плотный буфер обеспечивает хорошую защиту волокна от давления и ударов, кабель в плотном буфере имеет небольшой диаметр и допускает изгиб с относительно небольшим радиусом. Недостатком плотного буфера является чувствительность кабеля к изменению температуры.

В кабеле с *пустотельм* буфером (loose tube) волокна свободно располагаются в полости буфера – жесткой пластиковой трубке, а оставшееся пространство может быть заполнено гидрофобным гелем. Такая конструкция более громоздка, но обеспечивает большую устойчивость к растяжению и изменениям температуры. Здесь волокна имеют длину большую, чем длина кабеля, поэтому деформации оболочки не затрагивают само волокно.

Силовые элементы обеспечивают требуемую механическую прочность кабеля, принимая на себя растягивающие нагрузки. В качестве силовых элементов используют кевларовые нити, стальные стержни, стренги из

скрученной стальной проволоки, стеклопластиковые стержни. Самую высокую прочность имеет стальная проволока, но для полностью непроводящих кабелей она неприменима.

Внешняя оболочка защищает всю конструкцию кабеля от влаги, химических и механических воздействий. Кабели для тяжелых условий эксплуатации могут иметь многослойную оболочку, включающую и бронирующую рубашку из стальной ленты или проволоки. Материал внешней оболочки определяет защищенность кабеля от тех или иных воздействий, а также горючесть кабеля и токсичность выделяемого дыма.

В локальных сетях применяют кабели наружной, внутренней и универсальной прокладки. *Наружные* (outdoor) кабели отличаются лучшей защищенностью от внешних воздействий и более широким диапазоном допустимых температур. *Внутренний* (indoor) кабель, как правило, менее защищен, но и менее опасен при возгорании. *Универсальный* (indoor/outdoor) кабель сочетает в себе защищенность и безвредность, но, как правило, он дороже специализированного.

Распределительный (distribution) кабель состоит из множества волокон, его разделывают в распределительных коробках и панелях, корпуса которых защищают волокна от механических воздействий. Пример конструктивного выполнения кабеля показан на рис. 38.

Основные достоинства и недостатки:

1) высокая скорость передачи данных, что значительно повышает производительность сети;

2) высокая помехоустойчивость;

3) трудоемкость в обслуживании волоконно-оптического кабеля;

4) большие трудозатраты при установке и монтаже;

5) трудности при наращивании сети и создании ответвлений;

6) трудности при поиске неисправностей;

7) высокая стоимость.

Первоначально в качестве материала для изготовления ВОЛС использовалось стекло. В последнее время стали применять специальный прозрачный пластик.

Применительно к различным кабельным системам объединенный график, характеризующий производительность различных кабельных систем через скоростные характеристики, показан на рис. 39.

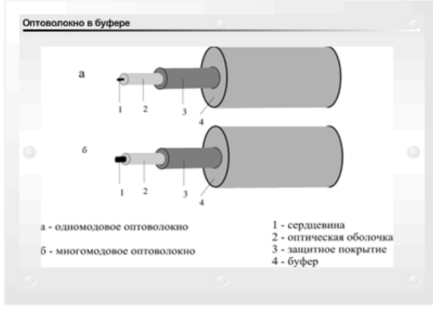

Рис. 38. Конструктивное выполнение оптоволоконного кабеля

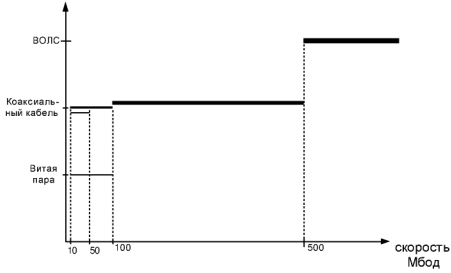

Рис. 39. Скоростные характеристики кабельных систем

3.2.1.4. Радиоканалы. Спутниковые каналы

Радиоканалы очень бурно развиваются в настоящее время. Для передачи данных используется радиосигнал или спутниковые каналы. Многие фирмы используют радиосигнал для дублирования оптоволокна. Особенностью спутниковых каналов является то, что стоимость не зависит от расстояния, на которое осуществляется передача. Компании, представляющие

услуги связи через спутниковые каналы, определяют цену в зависимости от времени и сеанса связи и от объема информации.

Спутниковая связь эффективна при условии, что данные передаются на расстояние более 800 км. В спутниковых системах используются антенны СВЧ-диапазона для приема сигнала с Земли и ретрансляции его на другую точку. На спутнике есть транспондер, который занимается передачей сигнала. Ретрансляция ведётся на другой частоте (чтобы не забивали друг друга). Сигналы со спутника могут быть приняты любой наземной станцией. Сигналы могут нести: речевые данные, данные, ТВ-сигналы (рис. 40).

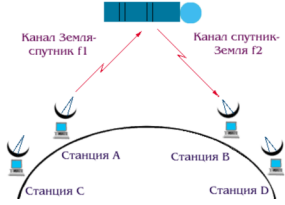

Рис. 40. Спутниковые каналы связи

Спутник служит электронной ретранслирующей станцией. Наземная станция А передает сигналы определенной частоты f1 (канал Земля-спутник). В свою очередь, спутник получает эти сигналы и ретранслирует их для приема наземной станцией В на частоте f2 канала спутник-Земля. Сигналы по этому каналу могут быть приняты любой станцией, которая находится в зоне приема. Сигналы могут нести речевую информацию, данные, телевизионные видеосигналы.

Спутник работает на частоте 1 ГГц. Большинство спутниковых каналов используют частоты 6/4 и 14/12 ГГц.

Частоты передачи и приема отличаются с целью предотвращения наложений и помех сигналов [1, 22].

3.3. Сигналы. Передача информации в компьютерных сетях посредством сигналов

Для передачи информации в ВС используются сигналы.

Сигналы – это физические процессы, параметры которых содержат информацию.

Назначение сигналов заключается в том, чтобы в каком-либо физическом процессе отобразить события, величины, функции. Для образования сигналов с целью передачи информации используются следующие базовые сигналы или носители:

- сигнал фиксированного уровня – имеет постоянное значение в течение определённого промежутка времени (показан на рис. 41);

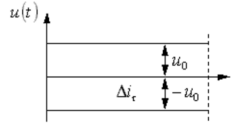

Рис. 41. Сигнал фиксированного уровня

- периодические сигналы (рис. 42);

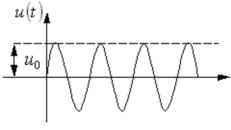

Рис. 42. Периодический сигнал

- импульсы любой физической природы (рис. 43).

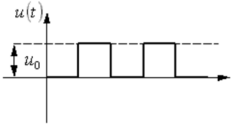

Рис. 43. Импульсный сигнал

В исходном состоянии физические носители этих трёх типов не несут никакой информации и представляют собой нечто вроде "чистой" поверхности, которую можно использовать для нанесения информации.

Нанесение информации достигается определенным изменением параметров физических процессов, состояний, соединений, колебательных элементов. Чаще всего для этих целей используется изменение параметров физических процессов, т. е. колебаний или импульсных последовательностей. Этот процесс называется модуляцией. Иными словами, можно сформулировать следующие определения.

Модуляция – операция, посредством которой осуществляется нанесение информации путем изменения параметров физических процессов, а именно, колебаний или импульсных последовательностей.

Демодуляция – обратная операция, посредством которой восстанавливается величина, вызвавшая изменение параметров при модуляции.

Для нанесения информации используются следующие параметры сигнала:

1) фиксированный уровень:
   а) уровень сигнала,
   б) полярность;
2) периодический сигнал:
   а) амплитуда,
   б) частота,
   в) фаза;
3) импульсы любой физической природы:
   а) уровень сигнала,
   б) длительность сигнала,
   в) фаза сигнала,
   г) частота,
   д) последовательность импульсов,
   е) скважность.

Математическое представление операции модуляции будет иметь следующий вид:

$U_1 = g[a_1, a_2, a_3, \ldots a_i, a_n, t]$ – немодулированный сигнал;

$U_2 = g[a_1, a_2, a_3, \ldots a_i + Dai(t), an, t]$ – модулированный сигнал,

где $Da_i(t)$ – переменная, составляющая параметра носителя, несущая информация, или модулирующая функция. Она связана с информативной или управляющей функцией линейной зависимостью $Da_i = kx$.

Рассматривая модуляцию, введем такое понятие, как информативные параметры. При модуляции можно изменять либо один параметр, либо несколько одновременно.

В зависимости от того, какое количество информативных параметров изменяется при нанесении информации, модуляция делится на простую и сложную.

Простая модуляция – это когда изменяется только один информативный параметр (a_i, i = 1).

Сложная модуляция – это когда изменяется некоторое количество параметров (a_i, i = 2).

Примеры простых видов модуляции для носителя второго типа гармонического сигнала приведены на рис. 44 и 45.

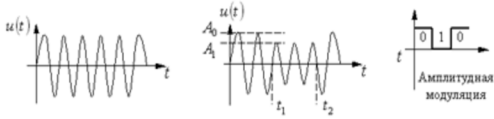

Рис. 44. Амплитудная модуляция

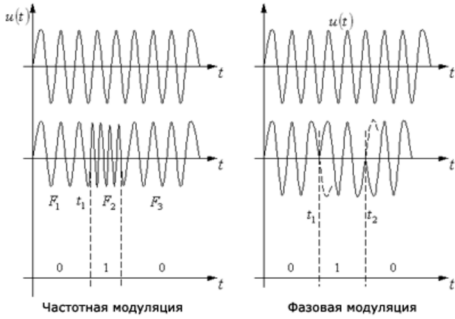

Рис. 45. Частотная и фазовая модуляция

Примеры простых видов модуляции для носителя третьего типа последовательности импульсов строгой периодичности показаны на рис. 46. К ним относятся следующие виды модуляции: амплитудно-импульсная (АИМ), частотно-импульсная (ЧИМ), времяимпульсная (ВИМ), фазоимпульсная (ФИМ), широтно-импульсная (ШИМ), счетно-импульсная (СЧИМ), кодоимпульсная модуляция (КИМ) [22].

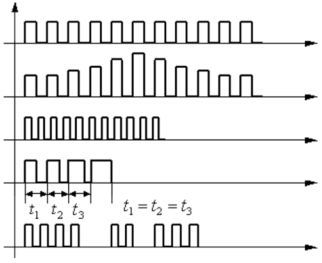

Рис. 46. Импульсная модуляция

3.3.1. Цифровое кодирование. Требования к методам цифрового кодирования

Цифровое кодирование информации

При цифровом кодировании дискретной информации применяются потенциальные и импульсные коды.

В потенциальных кодах для представления логических единиц и нулей используется только значение потенциала сигнала. Перепады, формирующие законченные импульсы, не используются.

Импульсные коды позволяют представить двоичные данные или импульсами определенной полярности, или частью импульсного перепада потенциала определенной полярности или определенного напряжения.

Требования к методам цифрового кодирования

При использовании прямоугольных импульсов для передачи дискретной информации необходимо выбрать такой способ кодирования, который одновременно достигал бы несколько целей:

- имел при одной и той же скорости наименьшую ширину спектра результирующего сигнала;
- обеспечивал синхронизацию между передатчиком и приемником;
- обладал способностью распознавать ошибки;
- обладал низкой стоимостью реализации.

Синхронизация передатчика и приемника

Синхронизация передатчика и приемника необходима, чтобы приемник точно знал, в какой момент времени необходимо считывать информацию, поступающую по линии связи. Данная проблема в вычислительных сетях решается значительно сложнее, чем при обмене данными между близкорасположенными устройствами.

На относительно небольших расстояниях хорошо зарекомендовала себя схема синхронизации, построенная с использованием отдельной тактирующей линии связи.

Неравномерное распространение сигнала может привести к тому, что тактовый импульс может прийти раньше или позже, в результате чего информация будет потеряна. Это является недостатком данной схемы синхронизации.

Выделенные дополнительные линии, как правило, являются дорогостоящими, что ведет в целом к увеличению стоимости данного способа (метода) передачи.

В вычислительных сетях используются самосинхронизирующиеся коды, которые несут информацию принимающему устройству о том, в какой момент времени необходимо производить считывание очередной порции информации (в какой момент времени вести распознавание очередного бита).

3.3.1.1. Схемы кодирования. Коды, используемые для передачи данных

Существуют следующие распространенные схемы двоичного кодирования:

1) Схема с использованием униполярного кодирования. При униполярном кодировании напряжение всех сигналов неотрицательно либо неположительно, т.е. алгебраический знак не меняется.

2) Полярное кодирование. Полярное кодирование предусматривает, что сигнал имеет положительный и отрицательный потенциалы. Противоположные алгебраические знаки образуют логическое состояние.

3) Биполярное кодирование. При биполярном кодировании изменение сигнала происходит между тремя уровнями.

AMI – плохосинхронизирующийся код.

Униполярное кодирование

Для кодирования двоичных единиц используются импульсы двоичной (одной) полярности.

Уровень сигнала при передаче данных посредством данного кода остается стабильным для каждой последовательности одноименных битов.

Код NRZ широко используется вследствие своей простоты и низкой стоимости. Большим достоинством NRZ-кода является эффективное использование полосы частот. Поскольку он может представлять бит для каждого изменения сигнала.

Недостатком данного кода является отсутствие способности самосинхронизации, так как длинные серии следующих подряд логических единиц и нулей не приводят к изменениям сигнала в канале.

Вывод: может произойти рассогласование или дрейф таймера приемника по отношению к поступающему сигналу и несвоевременный опрос линии связи. В результате передатчик и приемник утратят синхронизацию.

Полярное кодирование

RZ-код предусматривает, что при представлении каждого бита сигнал меняется по меньшей мере один раз. Так как RZ-код обеспечивает изменение состояния для каждого бита, то этот код обладает хорошим свойством синхронизации.

Недостатком является то, что он требует двух переходов или изменения сигнала для каждого бита.

Вывод: RZ-код требует вдвое больше битовой скорости по сравнению с обычным кодом. Код данного типа используется в автоматизированных системах, построенных на базе локальных вычислительных сетей, а также при использовании и применении лазерной и оптоволоконной технологии.

Манчестерский код обеспечивает изменение состояния сигнала при представлении каждого бита. Манчестерский код как вид кода очень распространен, используется в настоящее время во многих системах передачи данных, в вычислительных сетях, автоматизированных системах и распределенных информационно-измерительных системах, а также в информационно-вычислительных комплексах.

Вывод: Это очень хороший синхронизирующий код. Как и RZ-код, манчестерский код требует удвоенной скорости передачи данных для передачи заданного количества битов.

Биполярный код – при представлении логической единицы использует импульсы различной полярности. При передаче данных с использованием этого кода в передаваемом сообщении содержатся длинные последовательности логического нуля.

Вывод: Элементы систем или сетей передачи данных, вычислительные сети при использовании данного кода лишены возможности синхронизировать с нулевыми битами, вследствие того, что имеет место отсутствие изменения состояния линии связи.

Потенциальный код 2B1Q – это код с четырьмя уровнями сигнала для кодирования данных. Название кода отражает его суть – каждые 2 бита (2B) передаются за один такт сигналом, имеющим четыре состояния (1Q). Паре битов 00 соответствует потенциал +2,5 В, паре битов 01 соответствует потенциал + 0,833 В, паре 11– потенциал +0,833 В, а паре 10 – потенциал +2,5 В. При этом способе кодирования требуются дополнительные меры по борьбе с длинными последовательностями одинаковых пар битов, так как сигнал при этом превращается в постоянную составляющую. При случайном чередовании битов спектр сигнала в два раза уже, чем у кода NRZ, так как при той же битовой скорости длительность такта увеличивается в два раза.

Вывод: С помощью кода 2B1Q можно по одной и той же линии передавать данные в два раза быстрее, чем с помощью кода AMI или NRZI. Однако для его реализации мощность перадатчика должна быть выше, чтобы четыре уровня четко различались между собой.

Пример цифровых кодов получивших наиболее широкое распространение для передачи данных в сетях показан на рис. 47 [22].

3.3.2. Методы (режимы) передачи данных. Элементы передачи данных

При рассмотрении и анализе телекоммуникационных систем, к которым относятся сети передачи данных и компьютерные сети, можно выделить четыре базовых компонента (элемента):

Передающее устройство (передатчик). Это устройство предназначено для передачи данных, в качестве которого может выступать терминал или ЭВМ совместно с аппаратурой передачи данных. Передающее устройство передает или генерирует данные, предназначенные для передачи.

Сообщение. Цифровые данные или данные в аналоговом виде, которые передаются от передающего устройства к приемному. В качестве сообще-

ния может выступать какой-либо файл, ответ на запрос, файл базы данных и т.д.

Средства передачи. Это та среда, которая используется для передачи информации от передающего устройства к принимающему.

Приемное устройство. Устройство, предназначенное для приема данных, в качестве которого может выступать терминал, ЭВМ или какое-либо другое устройство.

Сеть передачи данных представим в виде структурной схемы, приведенной на рис. 48.

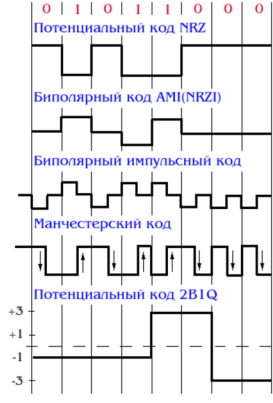

Рис. 47. Примеры цифровых кодов

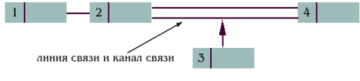

Рис. 48. Компоненты сети передачи данных

ООД и АКД посылают друг другу коммуникационный трафик с использованием одного из трех методов (режимов передачи):

1) **симплексного** – передача только в одном направлении;

2) **полудуплексного** – передача в обоих направлениях, но одновременно только в одном направлении (также называемого поочередно двунаправленным);

3) **дуплексного (или полнодуплексного)** – одновременная передача в обоих направлениях (также называемого одновременно двунаправленным).

Дополнительно к методам передачи данных относятся:

- параллельная и последовательная передача данных;
- асинхронная и синхронная передача.

Три раскрываемых ниже режима определяют направление передачи данных.

3.3.2.1. Симплексный режим

При симплексном режиме данные передаются только в одном направлении (рис.49). Используя транспортную аналогию, симплексную передачу можно представить как однонаправленную однополосную дорогу. Транспорт движется только в одну сторону и в один ряд.

Данный метод не так распространен в области передачи данных вследствие однонаправленного характера процесса, однако симплексные системы находят применение в некоторых областях науки таких, как телеметрия. Также симплексный метод широко используется в телевидении и коммерческом радиовещании.

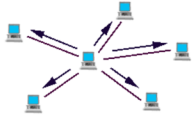

Рис. 49. Симплексный режим

3.3.2.2. Полудуплексный режим

Передача при данном режиме осуществляется в обоих направлениях, но одновременно только в одном направлении (так называемый поочередно двунаправленный). Полудуплексный режим является самым распространенным.

Данный режим похож на однополосную дорогу, по которой движение может осуществляться в обоих направлениях, но не одновременно, а последовательно (рис. 50).

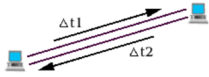

Рис. 50. Полудуплексный режим

Полудуплексный метод применяется во многих системах таких, как запросно-ответные прикладные системы, в которых некоторое устройство ООД посылает запрос другому устройству ООД и ожидает, что прикладной процесс (ПП) осуществит и/или вычислит ответ и передаст его назад. В системах на основе терминалов часто используются полудуплексные методы.

3.3.2.3. Дуплексный режим

Дуплексный (или полнодуплексный) режим – одновременная передача в обоих направлениях (так называемый одновременно двунаправленный). Данный режим похож на двухполосную, двунаправленную дорогу (рис. 51).

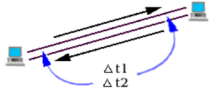

Рис. 51. Дуплексный режим

Этот метод обеспечивает одновременно двунаправленную передачу, при этом не возникают нежелательные остановки и ожидания, характерные для полудуплексного метода. Полнодуплексный метод широко используется в приложениях, требующих непрерывного использования канала, высокой производительности и быстрого ответа.

Асинхронный и синхронный способы передачи. Параллельная и последовательная передача.

Асинхронный способ передачи.

Асинхронный способ называется стартстопной передачей. При асинхронной передаче данные передаются в виде 0 и 1, поэтому приемник дол-

жен уметь выделять байты в этом потоке данных. Для этого каждый передаваемый байт обрабатывается стартовым и стоповым битом.

В отдельных случаях на низконадежных линиях связи разрешается использовать несколько таких битов, что приводит к дополнительным накладным расходам, аппаратным затратам и в целом снижает эффективность передачи.

Достоинство: способ передачи относительно недорогой, не требующий дорогостоящего оборудования.

Синхронный способ передачи.

Синхронная передача данных более быстрая, чем асинхронная, и при передачи информации информационные блоки не разделены стартовыми и стоповыми блоками. Передаваемые информационные блоки обрамляются специальными управляющими символами. Другие символы несут дополнительную информацию о данных и обеспечивают функции обнаружения ошибок.

Достоинство: более быстрый способ передачи данных и почти безошибочный. При синхронной передаче данные могут передаваться большими блоками.

Параллельный способ передачи данных.

При параллельном способе передачи группа битов передается одновременно по нескольким линиям, образующих магистраль, причем каждый бит передается по собственной линии. Параллельный способ передачи данных имеет смысл при передаче информации на короткие расстояния.

Достоинство: скорость передачи данных высокая.

Недостаток:

- ограничение на расстояние;
- взаимовлияние сигналов, распространяющихся по различным проводникам и линиям.

Последовательный способ передачи данных.

Достоинство: экономически выгодно информацию передавать на большие расстояния.

Недостаток: более медленный способ передачи данных [22].

Контрольные вопросы к разделу 3

1. Что лежит в основе построения линии связи (ЛС)?
2. Приведите классификацию ЛС.

3. Перечислите основные особенности проводных ЛС.
4. Перечислите основные особенности кабельных ЛС.
5. Перечислите основные характеристики ЛС.
6. Что характеризует АХЧ?
7. Что характеризует полоса пропускания или ширина полосы частот?
8. Приведите данные по диапазонам частот, используемым для передачи данных в ВС и СПД.
9. Чем характеризуется пропускная способность ЛС?
10. От чего зависит пропускная способность ЛС?
11. Приведите стандарты, регламентирующие требования, предъявляемые к кабельным системам ВС?
12. Кабельные системы на основе коаксиального кабеля (КС). Приведите основные характеристики КС.
13. Приведите примеры типов кабельных систем на базе КС.
14. Перечислите достоинства и недостатки кабельных систем на базе КС.
15. Кабельные системы на основе витой пары (ВП). Классификация ВП.
16. Перечислите примеры стандартов на неэкранированную ВП (UTP). Основные характеристики.
17. Перечислите примеры стандартов на экранированную ВП (STP). Основные характеристики.
18. Перечислите основные характеристики кабельных систем на базе STP.
19. Стандарт фирмы IBM на STP.
20. Волоконно-оптические линии связи (ВОЛС). Классификация рабочей среды – оптоволокна ВОЛС.
21. На какие показатели ВОЛС оказывает влияние дисперсия? Виды дисперсии.
22. Как классифицируется волокно в зависимости от режима распространения луча?
23. Перечислите достоинства и недостатки ВОЛС.
24. Радиоканалы, спутниковые каналы. Достоинства и недостатки.
25. Дайте определение, что такое сигнал.
26. Перечислите типы базовых сигналов носителей, используемых для формирования сигналов и передачи информации в ВС.
27. В чем заключается операция модуляции и демодуляции?
28. Понятие «информативные параметры сигналов». Перечислите информативные параметры для носителя первого типа.

29. Понятие «информативные параметры сигналов». Перечислите информативные параметры для носителя второго типа.
30. Понятие «информативные параметры сигналов». Перечислите информативные параметры для носителя третьего типа.
31. Укажите отличие простой модуляции от сложной.
32. Приведите пример простых видов модуляции для носителя второго типа – гармонический сигнал. Представьте в виде временной диаграммы.
33. Приведите пример сложных видов модуляции для носителя второго типа – гармонический сигнал. Представьте в виде временной диаграммы.
34. Приведите примеры простых видов модуляции для носителя третьего типа – импульсная последовательность.
35. Приведите примеры простых видов модуляции для носителя третьего типа – импульсная последовательность. Представьте в виде временной диаграммы.
36. Какие требования предъявляются к цифровому кодированию в ВС?
37. Перечислите основные схемы кодирования, применяемые в ВС.
38. NRZ-код: достоинства и недостатки при передаче данных. Временная диаграмма.
39. RZ-код: достоинства и недостатки при передаче данных. Временная диаграмма.
40. Манчестерский код: достоинства и недостатки при передаче данных. Временная диаграмма.
41. AMI код: достоинства и недостатки при передаче данных. Временная диаграмма.
42. 2B1Q код: достоинства и недостатки при передаче данных. Временная диаграмма.
43. Перечислите основные режимы (способы) передачи данных в ВС.
44. Симплексный режим передачи данных в ВС. Области применения. Достоинства и недостатки.
45. Полудуплексный режим передачи данных в ВС. Области применения. Достоинства и недостатки.
46. Дуплексный (полнодуплексный) режим передачи данных в ВС. Области применения. Достоинства и недостатки.
47. Синхронный способ обмена данными в ВС. Достоинства и недостатки.

48. Асинхронный способ обмена данными в ВС. Достоинства и недостатки.
49. Параллельный и последовательный способы обмена данными. Области применения. Достоинства и недостатки.
50. Структура телекоммуникационной системы. Перечислите основные компоненты.

4. ПРИМЕРЫ РЕАЛИЗАЦИИ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

Для организации взаимодействия прикладных процессов в современных вычислительных, компьютерных сетях различного уровня и масштаба необходимо использовать сетевые протоколы [14, 15, 16, 17].

Определение 1. Протоколом называется набор правил для одной из коммуникационных функций.

Определение 2. Протокол – это набор синтаксических и семантических правил, в соответствии с которыми осуществляется взаимодействие процессов одного уровня в сети.

Определение 3. Стек протоколов представляет собой набор организованных по уровням протоколов, которые, работая совместно, позволяют приложениям обмениваться данными.

Определение 4. Набор протоколов – это семейство протоколов, работающий совместно и связанных между собой. Набор протоколов TCP/IP обеспечивает множество различных возможностей, начиная от динамического определения адреса сетевого адаптера и заканчивая службой каталогов, определяющей способ доставки сообщения электронной почты.

Определение 5. Хостом называется компьютер, который выполняет приложения и имеет одного или нескольких пользователей. Поддерживающий TCP/IP хост, работает как конечная точка сетевой коммуникации.

Пример создания клиент-серверного приложения на основе потокового сокета.

Для начала разработки подключите к проекту библиотеку **ws2_32.lib** (в MVC++12.0: Project -> Settings -> Link -> Object/library modules, необходимо обязательно в конце строки дописать). Подключите используемый файл winsock2.h:

```
#include <winsock2.h>
```

До начала использования функций WinSock необходимо выполнить инициализацию WinSock.

```
WSADATA WSAData; // После инициализации содерж инф.
o WinSock
if (WSAStartup (MAKEWORD(2,1), &WSAData) != 0)
{
```

```
  printf ("WSAStartup faild. Error: %d",
WSAGetLastError ());
  return FALSE;
}
```

Приложение СЕРВЕР

Откройте сокет при помощи **socket** функции.

Используйте **AF_INET** (семейство интернет протоколов) для формата адреса (address format) и **SOCK_STREAM** (потоковый сокет) для типа используемого сокета (type parameter.)

Пример открытия сокета.

```
  int WinSocket;
if ((WinSocket = socket (AF_INET, SOCK_STREAM, 0)) ==
INVALID_SOCKET)
{
  printf ("Allocating socket failed. Error: %d", WSA-
GetLastError ());
  return -1;
}
```

Установление соединения с клиентом **accept** функцией. Потоковый сокет TCP использует функцию **accept** для приема запроса на соединение от клиента и именования данного соединения между клиентом и сервером. Функция **accept** создает новый сокет. Исходный сокет используется сервером для прослушивания запросов на соединение от клиентов.

```
  printf("Waiting connect from client. \n");
int ClientSock;
accept_sin_len = sizeof (accept_sin);
// Accept an incoming connection attempt on WinSocket.
ClientSock = accept (WinSocket,
                        (struct sockaddr *)
&accept_sin,
                        (int *) &accept_sin_len);
// Stop listening for connections from clients.
closesocket (WinSocket);
```

```
if (ClientSock == INVALID_SOCKET)
{
  printf ("Accepting connection with client failed.
Error: %d", WSAGetLastError ());
  return FALSE;
}
```

4.1. Программирование протоколов TCP, UDP, SPX, IPX

4.1.1. Описание основных функций управления протоколами

4.1.1.1. Функция WSAStartup

Функция **WSAStartup** инициализирует библиотеку WS2_32.DLL для использования процессом.

```
int WSAStartup(

  WORD wVersionRequested,

  LPWSADATA lpWSAData

);
```

wVersionRequested – Наивысшая версия Windows Sockets, которую вызвавший может использовать.

lpWSAData – Указатель на память структуры **WSADATA** для вывода подробной информации о Windows Sockets.

Функция **WSAStartup** возвращает ноль при удачном выполнении. Иначе, возвращает код ошибки.

Приложение должно вызывать одну функцию **WSACleanup** для каждой удачной функции **WSAStartup**.

4.1.1.2. Функция WSACleanup

Функция **WSACleanup** отключает применение библиотеки WS2_32.DLL.

```
int WSACleanup(void);
```

Функция не имеет параметров.

Возвращаемое значение ноль, если операция удалась. Иначе, возвращаемое значение равно SOCKET_ERROR.

4.1.1.3. Функция getaddrinfo

Функция **getaddrinfo** управляет протокольно-независимой трансляцией из имени хоста к адресу.

```
int getaddrinfo(
  const char* nodename,
  const char* servname,
  const struct addrinfo* hints,
  struct addrinfo** res
);
```

nodename – указатель на строку, содержащую имя хоста или его IP- адрес.

servname – указатель на строку, содержащую имя сервиса или номер порта.

hints – указатель на структуру **addrinfo**, которая содержит информацию о совместимом типе сокета.

res – указатель на список одной или нескольких структур **addrinfo**, содержащих полную информацию о хосте.

При удачном выполнении возвращает ноль. Иначе возвращает ненулевое значение, т.е. код ошибки.

4.1.1.4. Функция freeaddrinfo

Функция **freeaddrinfo** освобождает адресное пространство, которое функция **getaddrinfo** динамически заняла для структуры **addrinfo**.

```
void freeaddrinfo(
  struct addrinfo* ai
);
```

ai – указатель на список одной или нескольких структур **addrinfo** для освобождения.

Эта функция не возвращает значений.

4.1.1.5. Функция socket

Функция **socket** создает сокет, который прикреплен к специальному сервису.

```
SOCKET socket(

  int af,

  int type,

  int protocol

);
```

af – семейство адресов.

type – тип для нового сокета. SOCK_STREAM – логический канал или SOCK_DGRAM – дейтаграммы.

protocol – протокол, который будет использован с сокетом, определенный в семействе адресов.

Если без ошибок, функция **socket** вернет номер нового сокета. Иначе, значение, равное INVALID_SOCKET будет возвращено.

4.1.1.6. Функция bind

Функция **bind** закрепляет локальный адрес с сокетом.

```
int bind(

  SOCKET s,

  const struct sockaddr* name,

  int namelen

);
```

s – идентификатор незакрепленного сокета.

name – адрес, к которому будет закреплен сокет из структуры **sockaddr**.

namelen – длина значения в параметре name, в байтах.

Если нет ошибок, функция **bind** вернет ноль. Иначе она вернет SOCKET_ERROR.

4.1.1.7. Функция closesocket

Функция **closesocket** закрывает существующий сокет.

```
int closesocket(
  SOCKET s
);
```

s – сокет для закрытия.

Если нет ошибок, функция **closesocket** вернет ноль. Иначе она вернет значение, равное SOCKET_ERROR.

4.1.1.8. Функция listen

Функция **listen** помещает сокет в состояние, в котором он ожидает входящие соединения.

```
int listen(
  SOCKET s,
  int backlog
);
```

s – закрепленный и несоединенный сокет.

backlog – максимальное число входящих подключений. SOMAXCONN – максимальное значение.

Если нет ошибок, функция **listen** вернет ноль. Иначе она вернет значение, равное SOCKET_ERROR.

4.1.1.9. Функция connect

Функция **connect** осуществляет соединение к сокету.

```
int connect(
  SOCKET s,
  const struct sockaddr* name,
  int namelen
);
```

s – несоединенный сокет.

name – имя сокета в структуре **sockaddr,** с которым будет соединение.

namelen – длина name, в байтах.

Если нет ошибок, функция **connect** вернет ноль. Иначе она вернет значение, равное SOCKET_ERROR.

4.1.1.10. Функция recv

Функция **recv** принимает пакеты из соединенного или закрепленного сокета.

```
int recv(
  SOCKET s,
  char* buf,
  int len,
  int flags
);
```

s – соединенный сокет.

buf – буфер для входящих пакетов.

len – длина buf, в байтах.

flags – флаги, определяющие работу (желательно ставить 0).

Если нет ошибок, функция **recv** вернет число принятых байт. Если соединение было закрыто, то функция вернет ноль. В других случаях вернет SOCKET_ERROR.

4.1.1.11. Функция recvfrom

Функция **recvfrom** принимает дейтаграммы и значение адреса.

```
int recvfrom(
  SOCKET s,
  char* buf,
  int len,
  int flags,
  struct sockaddr* from,
  int* fromlen
);
```

s – закрепленный сокет.

buf – буфер для входящих пакетов.

len – длина buf, в байтах.

flags – флаги, определяющие работу (желательно ставить 0).

from – указатель на структуру **sockaddr**, в которой содержится информация об отправителе.

fromlen – указатель на размер from, в байтах.

Если нет ошибок, функция **recvfrom** вернет число принятых байт. Если соединение было закрыто, то функция вернет ноль. В других случаях вернет SOCKET_ERROR.

4.1.1.12. Функция send

Функция **send** отправляет пакеты соединенному сокету.

```
int send(
  SOCKET s,
  const char* buf,
  int len,
```

```
  int flags
);
```

s – соединенный сокет.

buf – буфер для приема пакетов.

len – длина buf, в байтах.

flags – флаги, определяющие работу (желательно ставить 0).

Если нет ошибок, функция **send** вернет число отправленных байт, которое может быть меньше чем значение len. В других случаях вернет SOCKET_ERROR.

4.1.1.13. Функция sendto

Функция **sendto** отправляет пакеты в определенное местоположение.

```
int sendto(
  SOCKET s,
  const char* buf,
  int len,
  int flags,
  const struct sockaddr* to,
  int tolen
);
```

s – идентификатор сокета (возможно соединенного).

buf – буфер, содержащий пакеты для отправки.

len – длина buf, в байтах.

flags – флаги, определяющие работу (желательно ставить 0).

to – указатель на структуру **sockaddr**, в которой содержится информация о получателе.

tolen – размер to, в байтах.

Если нет ошибок, функция **send** вернет число отправленных байт, которое может быть меньше чем значение len. В других случаях вернет SOCKET_ERROR.

4.1.1.14. Функция accept

Функция **accept** ожидает входящих соединений для сокета.

```
SOCKET accept(
  SOCKET s,
  struct sockaddr* addr,
  int* addrlen
);
```

s – прослушиваемый сокет.

addr – указатель на структуру **sockaddr**, в которой содержится информация о клиенте.

addrlen – размер addr, в байтах.

Если нет ошибок, функция **accept** вернет значение типа SOCKET для нового сокета, связанного с соединением.

4.1.2. Описание основных структур управления протоколами

4.1.2.1. Структура WSADATA

Структура **WSADATA** содержит информацию о Windows Sockets параметрах.

```
typedef struct WSAData {
  WORD wVersion;
  WORD wHighVersion;
  char szDescription[WSADESCRIPTION_LEN+1];
  char szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short iMaxSockets;
  unsigned short iMaxUdpDg;
  char FAR* lpVendorInfo;
```

```
} WSADATA, *LPWSADATA;
```

wVersion – версия Windows Sockets библиотеки Ws2_32.dll для использования.

wHighVersion – наивысшая версия Windows Sockets библиотеки .dll. Нормально если совпадает с **wVersion**.

szDescription – строка с кратким описанием библиотеки Ws2_32.dll.

szSystemStatus – строка, содержащая состояние библиотеки Ws2_32.dll.

iMaxSockets – это значение игнорируется (оставлено для старых версий).

iMaxUdpDg – это значение игнорируется (оставлено для старых версий).

lpVendorInfo – это значение игнорируется (оставлено для старых версий).

4.1.2.2. Структура addrinfo

Структура **addrinfo** используется для функции **getaddrinfo** для получения информации о хост- адресе.

```
typedef struct addrinfo {
  int ai_flags;
  int ai_family;
  int ai_socktype;
  int ai_protocol;
  size_t ai_addrlen;
  char* ai_canonname;
  struct sockaddr* ai_addr;
  struct addrinfo* ai_next;

} addrinfo;
```

ai_flags – флаги для функции **getaddrinfo**. AI_PASSIVE, AI_CANONNAME и AI_NUMERICHOST.

ai_family – семейство протоколов.

ai_socktype – тип сокета. SOCK_RAW, SOCK_STREAM, или SOCK_DGRAM.

ai_protocol – тип протокола.

ai_addrlen – длина **ai_addr**, в байтах.

ai_canonname – каноническое имя хоста.

ai_addr – указатель на структуру **sockaddr**.

ai_next – указатель на следующую структуру **addrinfo**. Если NULL, то структура **addrinfo** была последней в списке.

4.1.2.3. Структура sockaddr

Структура **sockaddr** содержит информацию о хосте.

```
struct sockaddr {

      u_short sa_family;

      char    sa_data[14];

};
```

sa_family – семейство протоколов.

sa_data – информация о хосте.

4.1.3. Использование протоколов

При программировании протоколов необходимо подключить библиотеку WS2_32.DLL при помощи функции **WSAStartup.** Затем получаем информацию об интересующем адресе функцией **getaddrinfo,** если мы используем протоколы семейства TCP/IP. Создаем сокет с параметрами нашего протокола с помощью функции **socket.**

Если мы используем сервер, то созданный сокет необходимо закрепить за определенным локальным адресом при помощи функции **bind**, затем вызвать функцию **listen** для этого сокета с заданным количеством соединений, если мы используем логический канал. Вызываем функцию **accept,** которая ожидает соединения с сервером и при входящем подключении создает новый сокет для логического канала. Использовать функцию **recv** для полу-

чения входящих пакетов логического канала или **recvfrom** для дейтаграмм. При использовании логического канала мы можем с помощью сервера отослать пакеты клиенту первыми через функцию **send**, а при дейтограммах нам необходимо дождаться пакет от клиента, чтобы определить его адрес для функции **sendto**.

При использовании клиента, после создания сокета, мы можем сразу подключиться к серверу функцией **connect** и начать передачу и прием пакетов для логического канала и дейтаграмм с использованием функций **send** и **recv**.

После завершения работы с сокетами, приложение должно отключить библиотеку WS2_32.DLL, используя функцию **WSACleanup** столько раз, сколько она была подключена.

4.1.4. Примеры фрагмента программ

4.1.4.1. Пример сервера с протоколом TCP

Простейший сервер использует локальный адрес и порт с номером 1212. Его задачей является ожидание подключения клиента и приём от него пакетов длиной 512, после приема каждого пакета отсылать строку “OK!!!” в качестве подтверждения.

```
  //---------------------------------------------------
-------------------------
  #include <stdio.h>
  #include <windows.h>
  #include <ws2tcpip.h>
  #pragma hdrstop

  //---------------------------------------------------
-------------------------

  #pragma argsused
  int main()
  {
    ADDRINFO f_hints, *f_addrinfo;
    SOCKET f_socket;
    memset(&f_hints, 0, sizeof(ADDRINFO));
    f_hints.ai_family=PF_INET;
```

```
    f_hints.ai_socktype=SOCK_STREAM;
    f_hints.ai_flags=AI_NUMERICHOST | AI_PASSIVE;

    WSADATA f_wsadata;
    if(WSAStartup(MAKEWORD(2, 2), &f_wsadata)!=0) re-
turn 0;

    if(getaddrinfo("0.0.0.0", "1212", &f_hints,
&f_addrinfo)!=0) return 0;
    if(f_addrinfo->ai_family!=PF_INET && f_addrinfo-
>ai_family!=PF_INET6)
    {
        freeaddrinfo(f_addrinfo);
        return 0;
    }
    f_socket=socket(f_addrinfo->ai_family, f_addrinfo-
>ai_socktype, f_addrinfo->ai_protocol);
    if(f_socket==INVALID_SOCKET)
    {
        freeaddrinfo(f_addrinfo);
        return 0;
    }
    if(bind(f_socket, f_addrinfo->ai_addr, f_addrinfo-
>ai_addrlen)==SOCKET_ERROR
    || listen(f_socket, 64)==SOCKET_ERROR)
    {
        closesocket(f_socket);
        freeaddrinfo(f_addrinfo);
        return 0;
    }
    freeaddrinfo(f_addrinfo);
    SOCKADDR_STORAGE f_from;
    int f_fromlen;
    while(1)
    {
        SOCKET f_socket2;
        f_fromlen=sizeof(f_from);
```

```
        f_socket2=accept(f_socket,
(LPSOCKADDR)&f_from, &f_fromlen);
        if(f_socket2==INVALID_SOCKET) break;

        char f_data[512];
        while(1)
        {
          memset(f_data, 0, 512);
          int f_size=recv(f_socket2, f_data, 512, 0);
          if(f_size<1) break;
          printf("%s/n", f_data);
          send(f_socket2, "OK!!!", 6, 0);
        }
        closesocket(f_socket2);
    }
    closesocket(f_socket);
    WSACleanup();
    return 0;
  }
  //-------------------------------------------------
-------------------------
```

4.1.4.2. Пример фрагмента программы клиента с протоколом TCP

Клиент соединяется с сервером и передает пакет с введенной строкой, затем высвечивает на экран строку с подтверждением “OK!!!” от сервера.

```
  //-------------------------------------------------
-------------------------
  #include <stdio.h>
  #include <windows.h>
  #include <ws2tcpip.h>
  #pragma hdrstop

  //-------------------------------------------------
-------------------------

  #pragma argsused
  int main()
```

```
  {
    ADDRINFO f_hints, *f_addrinfo;
    SOCKET f_socket;
    memset(&f_hints, 0, sizeof(f_hints));
    f_hints.ai_family=PF_INET;
    f_hints.ai_socktype=SOCK_STREAM;
    WSADATA f_wsadata;
    if(WSAStartup(MAKEWORD(2, 2), &f_wsadata)!=0) re-
turn 0;
    if(getaddrinfo("127.0.0.1", "1212", &f_hints,
&f_addrinfo)!=0) return 0;
    f_socket=socket(f_addrinfo->ai_family, f_addrinfo-
>ai_socktype, f_addrinfo->ai_protocol);
    if(f_socket==INVALID_SOCKET)
    {
        freeaddrinfo(f_addrinfo);
        return 0;
    }
    if(connect(f_socket, f_addrinfo->ai_addr,
f_addrinfo->ai_addrlen)==SOCKET_ERROR)
    {
        freeaddrinfo(f_addrinfo);
        closesocket(f_socket);
        return 0;
    }
    freeaddrinfo(f_addrinfo);
    char f_data[512];
    while(1)
    {
        gets(f_data);
        if(send(f_socket, f_data, strlen(f_data)+1,
0)!=SOCKET_ERROR)
        {
            recv(f_socket, f_data, 512, 0);
            puts(f_data);
        }
        else
```

```
        {
            WSACleanup();
            return 0;
        }
    }
    return 0;
}
//-------------------------------------------------
-------------------------
```

4.1.4.3. Комментарии к примерам

Для написания приложений с другими протоколами достаточно изменить их идентификаторы в функциях и ввести соответствующие адреса.

Текущие примеры используют файл включения ws2tcpip.h, который также используется для протокола UDP. Если вы собираетесь использовать протоколы SPX и IPX, то необходимо включить в программу файл wsipx.h, содержащий дополнительные структуры и описания.

Примеры написаны на Borland Developer Studio 2012. Для подробной справки используйте справочную систему вашего компилятора.

4.2. Программирование протокола NetBIOS

4.2.1. Описание основных функций управления протоколом

4.2.1.1. Функция Netbios

Функция **Netbios** предоставляет доступ к системам управления протоколом NetBIOS.

```
UCHAR Netbios(
      PNCB pncb
);
```

pncb – указатель на структуру **NCB,** которая описывает сетевые контрольные блоки.

Функция возвращает значение **ncb_retcode,** находящееся в структуре **NCB**.

4.2.2. Описание основных структур управления протоколом

4.2.2.1. Структура NCB

Структура **NCB** содержит информацию о командах, опциях и указателях на блоки памяти.

```
typedef struct _NCB {
      UCHAR ncb_command;
      UCHAR ncb_retcode;
      UCHAR ncb_lsn;
      UCHAR ncb_num;
      PUCHAR ncb_buffer;
      WORD ncb_length;
      UCHAR ncb_callname[NCBNAMSZ];
      UCHAR ncb_name[NCBNAMSZ];
      UCHAR ncb_rto;
      UCHAR ncb_sto;
      void (CALLBACK *ncb_post) (struct _NCB *);
      UCHAR ncb_lana_num;
      UCHAR ncb_cmd_cplt;
#ifdef _WIN64
      UCHAR ncb_reserve[18];
#else
```

```
    UCHAR ncb_reserve[10];

#endif

    HANDLE ncb_event;

} NCB, *PNCB;
```

ncb_command – команда для выполнения функцией **Netbios**.

ncb_retcode – возвращает результат после выполнения операции.

ncb_lsn – идентификатор номера сессии после удачного выполнения команды NCBCALL

ncb_num – содержит номер локального имени после удачного выполнения команды NCBADDNAME. Применяется для всех команд, использующих дейтаграммы.

ncb_buffer – указатель на буфер памяти.

ncb_length – размер буфера, в байтах.

ncb_callname – имя удаленного приложения.

ncb_name – имя, которое использует приложение для связи.

ncb_rto – тайм-аут для операции приема пакетов NCBRECV.

ncb_sto – тайм-аут для операции отсылки пакетов NCBSEND.

ncb_post – содержит адрес сообщения для вызова асинхронных команд.

ncb_lana_num – содержит номер LAN адаптера.

ncb_cmd_cplt – то же, что и **ncb_retcode.**

ncb_reserve – зарезервировано. Должно быть нулем.

ncb_event – содержит хендл объекта сообщения.

4.2.3. Использование протокола

В начале программы необходимо подготовить приложение к работе, используя команду NCBRESET с параметрами количества имен и сессий. Затем добавить новое имя в список локальных имен командой NCBADDNAME.

Если мы используем логический канал, то необходимо использовать команду NCBLISTEN для ожидания входящих подключений для сервера, а для клиента мы используем команду NCBCALL для подключения. После установки соединения по логическому каналу мы можем использовать команду NCBSEND для отсылки сообщений и команду NCBRECV для их приема. Для разрыва канала необходимо воспользоваться командой NCBHANGUP для сервера или клиента.

Если пользуемся дейтаграммами, то сразу для сервера используем команду приема сообщений NCBDGRECV, а для клиента команду передачи сообщения NCBDGSEND.

Для удаления неиспользуемого имени из списка возьмем команду NCBDELNAME, а для вывода списка имен команду NCBASTAT. LAN-адаптеры можно посмотреть с помощью команды NCBENUM.

4.2.4. Примеры программ

4.2.4.1. Пример фрагмента программы сервера с протоколом NetBIOS

Сервер использует LAN-адаптер с номером 0 и локальное имя "NewServer". Ожидает подключения клиента с любым именем, принимает от него пакеты размером 512 байт. После каждого принятого пакета отсылает клиенту строку "OK!!!". После выхода из приложения все имена текущего приложения автоматически удаляются из списка локальных имен. В данном примере рассмотрен сервер, использующий логический канал.

```
  //---------------------------------------------------
-------------------------
  #include <stdio.h>
  #include <windows.h>
  #include <nb30.h>
  #pragma hdrstop

  //---------------------------------------------------
-------------------------

  #pragma argsused
  int main()
  {
```

```
NCB f_ncb;
int Netbios_Lana=0;

memset(&f_ncb, 0, sizeof(NCB));
f_ncb.ncb_command=NCBRESET;
f_ncb.ncb_lsn=0;
f_ncb.ncb_callname[0]=20;
f_ncb.ncb_callname[2]=30;
f_ncb.ncb_lana_num=Netbios_Lana;
Netbios(&f_ncb);
if(f_ncb.ncb_retcode!=0) return 0;

    memset(&f_ncb, 0, sizeof(NCB));
f_ncb.ncb_command=NCBADDNAME;
memset(f_ncb.ncb_name, 0, NCBNAMSZ);
strcpy(f_ncb.ncb_name, "NewServer");
f_ncb.ncb_lana_num=Netbios_Lana;
Netbios(&f_ncb);
if(f_ncb.ncb_retcode!=0) return 0;

int F_Session=-1;

while(1)
{
    memset(&f_ncb, 0, sizeof(NCB));
    f_ncb.ncb_command=NCBLISTEN;
    memset(f_ncb.ncb_callname, 0, NCBNAMSZ);
    strcpy(f_ncb.ncb_callname, "*");
    memset(f_ncb.ncb_name, 0, NCBNAMSZ);
    strcpy(f_ncb.ncb_name, "NewServer");
    f_ncb.ncb_lana_num=Netbios_Lana;
    Netbios(&f_ncb);
    if(f_ncb.ncb_retcode!=0) return 0;
    F_Session=f_ncb.ncb_lsn;

    unsigned char Data[512];
```

```
        while(1)
        {
            memset(&f_ncb, 0, sizeof(NCB));
            f_ncb.ncb_command=NCBRECV;
            f_ncb.ncb_lsn=F_Session;
            f_ncb.ncb_buffer=Data;
            f_ncb.ncb_length=512;
            f_ncb.ncb_lana_num=Netbios_Lana;
            Netbios(&f_ncb);
            if(f_ncb.ncb_retcode!=0) break;

            puts(Data);

            memset(&f_ncb, 0, sizeof(NCB));
            f_ncb.ncb_command=NCBSEND;
            f_ncb.ncb_lsn=F_Session;
            f_ncb.ncb_buffer=(unsigned char*)"OK!!!";
            f_ncb.ncb_length=6;
            f_ncb.ncb_lana_num=Netbios_Lana;
            Netbios(&f_ncb);
        }
    }
    return 0;
}
//-------------------------------------------------
-------------------------
```

4.2.4.2. Пример фрагмента программы клиента с протоколом NetBIOS

Клиент использует LAN- адаптер 0 и локальное имя “NewClient”. Соединяется с сервером по логическому каналу и передает введенные строки, затем получает от сервера строку “OK!!!”. Также после выхода все локальные имена приложения удаляются.

```
//-------------------------------------------------
-------------------------
```

```
#include <stdio.h>
#include <windows.h>
#include <nb30.h>
#pragma hdrstop

//---------------------------------------------------
------------------------

#pragma argsused
int main()
{
   NCB f_ncb;
   int Netbios_Lana=0;

   memset(&f_ncb, 0, sizeof(NCB));
   f_ncb.ncb_command=NCBRESET;
   f_ncb.ncb_lsn=0;
   f_ncb.ncb_callname[0]=20;
   f_ncb.ncb_callname[2]=30;
   f_ncb.ncb_lana_num=Netbios_Lana;
   Netbios(&f_ncb);
   if(f_ncb.ncb_retcode!=0) return 0;

       memset(&f_ncb, 0, sizeof(NCB));
   f_ncb.ncb_command=NCBADDNAME;
   memset(f_ncb.ncb_name, 0, NCBNAMSZ);
   strcpy(f_ncb.ncb_name, "NewClient");
   f_ncb.ncb_lana_num=Netbios_Lana;
   Netbios(&f_ncb);
   if(f_ncb.ncb_retcode!=0) return 0;

   int F_Session=-1;

   memset(&f_ncb, 0, sizeof(NCB));
   f_ncb.ncb_command=NCBCALL;
   memset(f_ncb.ncb_callname, 0, NCBNAMSZ);
   strcpy(f_ncb.ncb_callname, "NewServer");
```

```
    memset(f_ncb.ncb_name, 0, NCBNAMSZ);
    strcpy(f_ncb.ncb_name, "NewClient");
    f_ncb.ncb_lana_num=Netbios_Lana;
    Netbios(&f_ncb);
    if(f_ncb.ncb_retcode!=0) return 0;
    F_Session=f_ncb.ncb_lsn;

    unsigned char Data[512];

    while(1)
    {
        gets(Data);
        memset(&f_ncb, 0, sizeof(NCB));
        f_ncb.ncb_command=NCBSEND;
        f_ncb.ncb_lsn=F_Session;
        f_ncb.ncb_buffer=Data;
        f_ncb.ncb_length=strlen(Data)+1;
        f_ncb.ncb_lana_num=Netbios_Lana;
        Netbios(&f_ncb);

        memset(&f_ncb, 0, sizeof(NCB));
        f_ncb.ncb_command=NCBRECV;
        f_ncb.ncb_lsn=F_Session;
        f_ncb.ncb_buffer=Data;
        f_ncb.ncb_length=512;
        f_ncb.ncb_lana_num=Netbios_Lana;
        Netbios(&f_ncb);
        if(f_ncb.ncb_retcode!=0) break;

        puts(Data);
    }
    return 0;
}
//-------------------------------------------------
-------------------------
```

4.2.4.3. Комментарии к примерам

Для построения приложения с дейтаграммами не нужно использовать команды прослушивания и вызова, достаточно просто начать прием дейтаграмм или их передачу. Также используются номера имен, а не их значения.

Примеры написаны на Borland Developer Studio 2012. Для подробной справки используйте справочную систему вашего компилятора.

4.3. Примеры реализации программ

Задание 1. Тип эксперимента: пакет.
Размер пакета: 32 – 512 байт.
Тайм-аут: 5 мс.
Вид соединения: дейтаграмма.
Размер передаваемых данных: 8 Кбайт.
Вывод статистики на сервере. Необходимо построить гистограмму, отображающую зависимость времени передачи от размера пакета.

Листинг программы. Реализация на языке C#

Модуль «Клиент»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace SIT
{
```

```
public partial class Form1 : Form
{
   private int localPort;                  // локальный порт
   private int remotePort;                 // удаленный порт
   private int timeOut;                    // тайм-аут
   private long size_data;                 // размер передаваемых данных
   IPAddress remoteIPAddres;               // удаленный IP- адрес
   public int[] sizes =                    // массив размеров отправляемых пакетов
      new int[] { 32, 64, 128, 256, 512 };
   public int s = 0;   // переменная для вывода на экран размера отправленного пакета
   private delegate void InvokeDelegate();   // делегат, для вывода информации
   private AutoResetEvent autoEvent = new AutoResetEvent(false);   // события для синхронизации потоков
   private AutoResetEvent autoEvent2 = new AutoResetEvent(false);

   void PrintLabelStart()
   {
      label2.Text = "подключение";
   }
   void PrintLabelConnect()
   {
      label2.Text = "подключено";
   }
   void PrintConnection()
   {
      richTextBox1.Text += DateTime.Now + " Подключение к серверу\n";
   }
   void PrintConnectionEnd()
   {
      richTextBox1.Text += DateTime.Now + " Соединение установленно\n";
      richTextBox1.SelectionStart = richTextBox1.Text.Length;
      richTextBox1.ScrollToCaret();
```

```
        }
        void PrintRequest()
        {
          richTextBox1.Text += DateTime.Now + " Отправлен запрос\n";
          richTextBox1.SelectionStart = richTextBox1.Text.Length;
          richTextBox1.ScrollToCaret();
        }
        void PrintPacked()
        {
          richTextBox1.Text += DateTime.Now + " Отправлен пакет размером " + s + " байт\n";
          richTextBox1.SelectionStart = richTextBox1.Text.Length;
          richTextBox1.ScrollToCaret();
        }
        void PrintAnsver()
        {
          richTextBox1.Text += DateTime.Now + " Получен ответ\n";
          richTextBox1.SelectionStart = richTextBox1.Text.Length;
          richTextBox1.ScrollToCaret();
        }
        void PrintEnd()
        {
          richTextBox1.Text += DateTime.Now + " Отправка данных завершена\n";
          richTextBox1.SelectionStart = richTextBox1.Text.Length;
          richTextBox1.ScrollToCaret();
        }
        void EnableButton()
        {
          button2.Enabled = true;
        }

        private void Recive()
        {
          try
          {
```

```
        UdpClient Recive = new UdpClient(localPort);
        IPEndPoint reciver = null;
        Recive.Receive(ref reciver);
        Recive.Close();
        autoEvent.Set();
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.ToString());
      }
    }
    private void ReciveData()
    {
      try
      {
        UdpClient Recive = new UdpClient(localPort);
        IPEndPoint reciver = null;
        while (true)
        {
          Recive.Receive(ref reciver);
          autoEvent2.Set();
        }
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.ToString());
      }
    }
    private void Request()
    {
      try
      {
        label2.BeginInvoke(new InvokeDelegate(PrintLabelStart));
        richTextBox1.BeginInvoke(new InvokeDelegate(PrintConnection));

        UdpClient Send = new UdpClient();
```

```
            IPEndPoint sender = new IPEndPoint(remoteIPAddres, remotePort);

            Thread tRec = new Thread(Recive);
            tRec.IsBackground = true;
            tRec.Start();

            string welcome = localPort.ToString() + "*" + textBox4.Text;
            byte[] data = Encoding.UTF8.GetBytes(welcome);

            while (true)
            {
               Send.Send(data, data.Length, sender);
               richTextBox1.BeginInvoke(new InvokeDelegate(PrintRequest));
               if (autoEvent.WaitOne(1000) = true)
               {
                  richTextBox1.BeginInvoke(new InvokeDele-
gate(PrintAnsver));
                  break;
               }
            }
            Send.Close();

            label2.BeginInvoke(new InvokeDelegate(PrintLabelConnect));
            richTextBox1.BeginInvoke(new InvokeDele-
gate(PrintConnectionEnd));
            button2.BeginInvoke(new InvokeDelegate(EnableButton));
         }
         catch(Exception ex)
         {
            MessageBox.Show(ex.ToString());
         }
      }
      private void SendData()
      {
         try
         {
```

```
            UdpClient Send = new UdpClient();
            IPEndPoint sender = new IPEndPoint(remoteIPAddres, remotePort);

            int index = 0;
            s = sizes[index];
            long tempSize = 0;

            Thread tRec2 = new Thread(ReciveData);
            tRec2.IsBackground = true;
            tRec2.Start();
            while (true)
            {
               byte[] Data = new byte[sizes[index]];
               Send.Send(Data, Data.Length, sender);
               richTextBox1.BeginInvoke(new InvokeDelegate(PrintPacked));
               if (autoEvent2.WaitOne(timeOut) = true)
               {
                  tempSize += sizes[index];
                  richTextBox1.BeginInvoke(new InvokeDele-
gate(PrintAnsver));
                  autoEvent2.Reset();
                  Thread.Sleep(50);
               }
               if (tempSize = size_data)
               {
                  if (index < (sizes.Length – 1))
                  {
                     index++;
                     s = sizes[index];
                     tempSize = 0;
                  }
                  else
                     break;
                  tempSize = 0;
               }
            }
```

```
            byte[] data = Encoding.UTF8.GetBytes("end");
            Send.Send(data, data.Length, sender);

            Send.Close();
            richTextBox1.BeginInvoke(new InvokeDelegate(PrintEnd));
          }
          catch(Exception ex)
          {
            MessageBox.Show(ex.ToString());
          }

        }
        public Form1()
        {
          InitializeComponent();
          try
          {
            // Получение имени компьютера.
            String host = System.Net.Dns.GetHostName();
            // Получение ip-адреса.
            System.Net.IPAddress ip = Sys-
tem.Net.Dns.GetHostByName(host).AddressList[0];
            this.textBox4.Text = ip.ToString();
          }
          catch(Exception ex)
          {
            MessageBox.Show(ex.ToString());
          }
        }
        private void button1_Click(object sender, EventArgs e)
        {
          try
          {
            localPort = int.Parse(textBox1.Text);
            remotePort = int.Parse(textBox2.Text);
            timeOut = int.Parse(textBox5.Text);
```

```
            remoteIPAddres = IPAddress.Parse(textBox3.Text);
            size_data = int.Parse(textBox6.Text);

            Thread tRec = new Thread(Request);
            tRec.IsBackground = true;
            tRec.Start();
          }
          catch(Exception ex)
          {
            MessageBox.Show(ex.ToString());
          }
        }
        private void button2_Click(object sender, EventArgs e)
        {
          try
          {
            Thread tRec = new Thread(SendData);
            tRec.IsBackground = true;
            tRec.Start();
          }
          catch(Exception ex)
          {
            MessageBox.Show(ex.ToString());
          }
        }

    }
}
```

Модуль «Сервер»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Diagnostics;

namespace SIT2
{
  public partial class Form1 : Form
  {
    private int localPort;              // локальный порт
    private int remotePort;             // удаленный порт
    IPAddress remoteIPAddres;           // удаленный IP-адрес
    int size;                           // размер принятого пакета
    private AutoResetEvent autoEvent = new AutoResetEvent(false);
    private delegate void InvokeDelegate();     // делегат, для вывода ин-
формации
    private int[] sizes =               // массив размеров отправляемых пакетов
      new int[] { 32, 64, 128, 256, 512 };
    private double[] times = new double[]{0, 0, 0, 0, 0};   // массив времен
    private Stopwatch sWatch = new Stopwatch();         // таймер

    void PrintConnectClient()
    {
      richTextBox1.Text += DateTime.Now + " Подключен клиент - " +
remoteIPAddres.ToString() + "\n";
      richTextBox1.SelectionStart = richTextBox1.Text.Length;
      richTextBox1.ScrollToCaret();
    }
    void PrintData()
    {
      richTextBox1.Text += DateTime.Now + " Получен пакет данных
размером " + size + " байт\n";
      richTextBox1.SelectionStart = richTextBox1.Text.Length;
```

```
            richTextBox1.ScrollToCaret();
        }
        void PrintEnd()
        {
            richTextBox1.Text += DateTime.Now + " Передача данных завер-
шена\n";
            richTextBox1.SelectionStart = richTextBox1.Text.Length;
            richTextBox1.ScrollToCaret();
        }
        void PrintChart()
        {
            chart1.Series["Series1"].Points.DataBindXY(sizes, times);
        }
        public Form1()
        {
            InitializeComponent();
            try
            {
                // Получение имени компьютера.
                String host = System.Net.Dns.GetHostName();
                // Получение ip-адреса.
                System.Net.IPAddress ip = System.Net.Dns.GetHostByName(host).
AddressList[0];
                this.textBox2.Text = ip.ToString();
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.ToString());

            }
        }
        void Connect()
        {
            try
            {
                UdpClient receivingUdpClient = new UdpClient(localPort);
```

```
            IPEndPoint RemoteIpEndPoint = null;

            byte[] data = receivingUdpClient.Receive(ref RemoteIpEndPoint);

            string infoClient = Encoding.UTF8.GetString(data);
            string port = "", ip = "";
            bool flag = false;
            for (int i = 0; i < infoClient.Length; ++i)
            {
               if (infoClient[i] == '*')
               {
                  flag = true;
                  continue;
               }

               if (!flag)
                  port += infoClient[i];
               else
                  ip += infoClient[i];
            }
            remotePort = int.Parse(port);
            remoteIPAddres = IPAddress.Parse(ip);
            UdpClient send = new UdpClient();
            IPEndPoint sendering = new IPEndPoint(remoteIPAddres, remote-
Port);
            byte[] datagram = new byte[8];
            send.Send(datagram, datagram.Length, sendering);

            receivingUdpClient.Close();
            send.Close();
            autoEvent.Set();
            richTextBox1.BeginInvoke(new InvokeDelegate
(PrintConnectClient));
         }
         catch(Exception ex)
         {
```

```
            MessageBox.Show(ex.ToString());
        }
    }
    void Reciving()
    {
        try
        {
            autoEvent.WaitOne();
            UdpClient receivingUdpClient = new UdpClient(localPort);
            IPEndPoint RemoteIpEndPoint = null;

            UdpClient send = new UdpClient();
            IPEndPoint sendering = new IPEndPoint(remoteIPAddres, remote-
Port);

            while (true)
            {
                sWatch.Start();
                byte[] data = receivingUdpClient.Receive(ref RemoteIpEnd-
Point);
                sWatch.Stop();
                string tempEnd = Encoding.UTF8.GetString(data);
                if (tempEnd == "end") break;
                size = data.Length;
                richTextBox1.BeginInvoke(new InvokeDelegate(PrintData));
                byte[] datagram = new byte[8];
                send.Send(datagram, datagram.Length, sendering);

                switch (data.Length)
                {
                    case 32:
                        times[0] += (double)sWatch.ElapsedMilliseconds / 1000;
                        break;
                    case 64:
                        times[1] += (double)sWatch.ElapsedMilliseconds / 1000;
                        break;
```

```
                case 128:
                    times[2] += (double)sWatch.ElapsedMilliseconds / 1000;
                    break;
                case 256:
                    times[3] += (double)sWatch.ElapsedMilliseconds / 1000;
                    break;
                case 512:
                    times[4] += (double)sWatch.ElapsedMilliseconds / 1000;
                    break;
            };
            sWatch.Reset();
        }
        receivingUdpClient.Close();
        send.Close();
        richTextBox1.BeginInvoke(new InvokeDelegate(PrintEnd));
        chart1.BeginInvoke(new InvokeDelegate(PrintChart));
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.ToString());
      }
    }
    private void button1_Click(object sender, EventArgs e)
    {
      richTextBox1.Text += DateTime.Now + " Ожидается подключение
клиента\n";
      try
      {
        localPort = int.Parse(textBox1.Text);
        Thread tRecQ = new Thread(Connect);
        tRecQ.IsBackground = true;
        tRecQ.Start();

        Thread tRecR = new Thread(Reciving);
        tRecR.IsBackground = true;
```

```
            tRecR.Start();
          }
          catch(Exception ex)
          {
            MessageBox.Show(ex.ToString());
          }
        }
      }
    }
```

Примеры результата выполнения программы показаны на рис. 52 и рис. 53.

Клиент

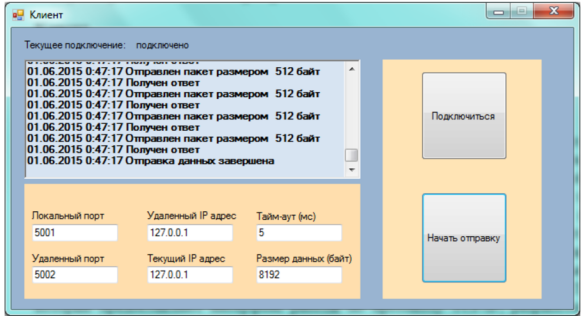

Рис. 52. Результат работы «Клиента»

Сервер

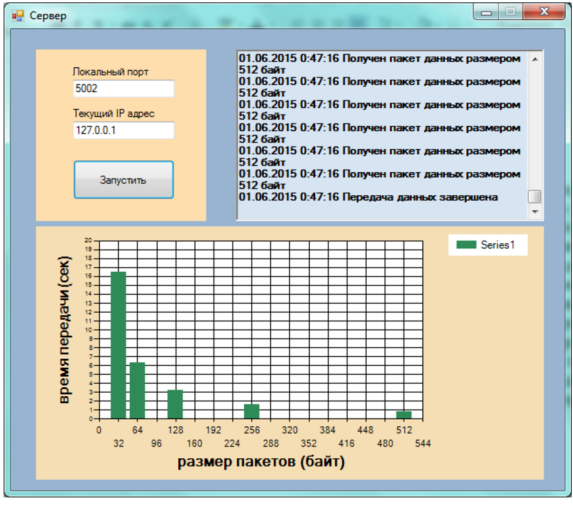

Рис. 53. Результат работы «Сервера»

Задание 2. Объем передаваемых данных: **16 Кбайт.**
Вид соединения: **Дейтаграмма.**
Таймаут: **15 мс.**

Листинг программы. Реализация на языке C++

```
#include<vcl.h>
#pragma hdrstop
#include <nb30.h>
#include <time.h>
#include "Unit1.h"
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TfMain *fMain;
TFileStream *inpFile, *outFile;
LANA_ENUM lan_number;
NCB ncb;
intWorkingLana=2, nSessions=20, nNames=30, TimeOut=15, count=0,
NBNAMENUM=0;
int s=0;
char NameString[]="Клиент", PriemString[]="Сервер";
float size=0;
int VOLUMES[]={8, 16, 32, 64, 128, 256, 512};
double TIMES[]={0,0,0,0,0,0,0};

intVolum=8;
intdeltaX=30;
#define B_S 512
#define TotalSize 24576
void Priem();

__fastcallTfMain::TfMain(TComponent* Owner)
      : TForm(Owner)
{
}

boolNbReset(byte l)
{
memset(&ncb,0,sizeof(ncb));
ncb.ncb_command=NCBRESET;
ncb.ncb_lana_num=l;
ncb.ncb_callname[0]=nSessions;
ncb.ncb_callname[2]=nNames;
  if(Netbios(&ncb)!=NRC_GOODRET)
  {
fMain->Memo->Lines->Add("Reset NCB Error. RetCode:
0x"+IntToStr(ncb.ncb_retcode));
    return 0;
```

```
    }
    else return 1;
   }

   void PaintStatistic()
   {
   int Max=TIMES[0];
    for (int r=1; r<7; r++) if(TIMES[r]>Max) Max=TIMES[r];
   int y;
   int x=0;
   fMain->Im->Canvas->Brush->Color=clWhite;
    fMain->Im->Canvas->Rectangle(0,0,fMain->Im->Width,fMain->Im-
>Height);

    for (int r=0; r<7; r++)
    {
     if (Max != 0) y = fMain->Im->Height - fMain->Im-
>Height*TIMES[r]/(Max*1.5);
      else y = 0;
   fMain->Im->Canvas->Brush->Color=clBlue;
   fMain->Im->Canvas->Rectangle(x,y,(x+deltaX),fMain->Im->Height);
   fMain->Im->Canvas->Brush->Color=clWhite;
   fMain->Im->Canvas->TextOutA(x+3,y-30,IntToStr(VOLUMES[r])+" байт");
   fMain->Im->Canvas->TextOutA(x+3,y-15,FloatToStr(TIMES[r])+" сек");
     x+=deltaX;
    }
   }

   void __fastcallTfMain::btFileClick(TObject *Sender)
   {
    if (OpenDialog->Execute())
    {
   edFile->Text=OpenDialog->FileName;
   inpFile = new TFileStream(edFile->Text, fmOpenRead);
     s=inpFile->Size;
   lbFileSize->Caption = IntToStr(s) + " байт";
```

```
    bt4->Enabled = true;
  }
}
void __fastcallTfMain::FormClose(TObject *Sender, TCloseAction&Action)
{
  delete inpFile;
}


void __fastcallTfMain::rbOtprClick(TObject *Sender)
{
edFile->Enabled=1;
btFile->Enabled=1;
lbFileSize->Enabled=1;
btRecv->Visible = 0;
  bt4->Visible = 1;
lbName->Caption="Имя: Клиент";
}

void __fastcallTfMain::rbPolClick(TObject *Sender)
{
edFile->Enabled=0;
btFile->Enabled=0;
lbFileSize->Enabled=0;
  bt4->Visible = 0;
btRecv->Visible = 1;
lbName->Caption="Имя: Сервер";
}

void __fastcallTfMain::btLANAClick(TObject *Sender)
{
  Memo->Lines->Add("----------------------------------------");
  if (NbReset(WorkingLana))
  {
memset(&ncb,0,sizeof(ncb));
ncb.ncb_command=NCBENUM;
```

```
   ncb.ncb_buffer=(unsigned char*)&lan_number;
   ncb.ncb_length=sizeof(lan_number);
      if(Netbios(&ncb)==NRC_GOODRET)
      {
        Memo->Lines->Add("LANA Number List:");
       for(inti=0; i<lan_number.length; i++)
         Memo->Lines->Add(IntToStr(i+1)+":
["+IntToStr(lan_number.lana[i])+"]");
        Memo->Lines->Add("----------------------------------------");
      }
      else
      {
       Memo->Lines->Add("LANA NCB Error. RetCode:
0x"+IntToStr(ncb.ncb_retcode));
      }
     }
   btGetLANA->Enabled=1;
   edGetLANA->Enabled=1;
   }


   void __fastcallTfMain::btRecvClick(TObject *Sender)
   {
   Priem();
   }

   char Buffer[B_S]={0};
   clock_t Begin=clock();
   double Tim=0;

   void Priem()
   {
    if(NbReset(WorkingLana))
    {
   TimeOut=StrToInt(fMain->edTimeOUT->Text);
   memset(&ncb,0,sizeof(ncb));
```

```
ncb.ncb_command=NCBADDNAME;
ncb.ncb_lana_num=WorkingLana;
ncb.ncb_rto = TimeOut;
memset(ncb.ncb_name, 0, NCBNAMSZ);
memcpy(ncb.ncb_name, PriemString, lstrlen(PriemString));
fMain->Refresh();
   NBNAMENUM=ncb.ncb_num;
   if(Netbios(&ncb)==NRC_GOODRET)
   {
      count=0;
      while (1)
      {
fMain->Refresh();
       char Buffer[B_S]={0};
       Buffer[B_S]=NULL;
       count++;
memset(&ncb,0,sizeof(ncb));
ncb.ncb_command=NCBDGRECV;
ncb.ncb_lana_num=WorkingLana;
ncb.ncb_num=0xFF;
ncb.ncb_buffer=(PUCHAR)Buffer;
ncb.ncb_length=Volum;
       if(Netbios(&ncb)==NRC_GOODRET)
       {
        if ( Buffer[1]=='~' && Buffer[2]=='~' && Buffer[3]==' ')
{
fMain->Memo->Lines->Add("Сообщение было получено");
fMain->Memo->Lines->Add(" ");
goto Exit;
         }
         if ( Buffer[1]=='~' && Buffer[2]==' ' && Buffer[3]==' ')
         {
Volum*=2;
          count=0;
fMain->Memo->Lines->Add("Next: "+IntToStr(Volum)+" байт");
fMain->Memo->Lines->Add(" ");
```

```
            }
            else
            {
   fMain->Memo->Lines->Add("Получено");
   fMain->Memo->Lines->Add(IntToStr(count)+": "+Buffer);
   fMain->Memo->Lines->Add(" ");
            }
           }
           Buffer[B_S]=NULL;
          } // while
      } // if Netbios
      else
      {
   fMain->Memo->Lines->Add("ADDNAME NCB Error. RetCode:
0x"+IntToStr(ncb.ncb_retcode));
   fMain->Memo->Lines->Add("----------------------------------------");
      }
    } // if Reset
    Exit:
   }

   inti=0;

   //---------------------------------------------------------------------------
   void __fastcallTfMain::bt4Click(TObject *Sender)
   {
   intSendData=0;
   Volum = VOLUMES[i];
   lbV->Caption="Packet: "+IntToStr(Volum)+" byte(s)";
   fMain->Refresh();
      char Buffer[B_S]={0};
      if (NbReset(WorkingLana))
      {
   TimeOut=StrToInt(edTimeOUT->Text);
   memset(&ncb,0,sizeof(ncb));
   ncb.ncb_command=NCBADDNAME;
```

```
ncb.ncb_lana_num=WorkingLana;
ncb.ncb_rto = TimeOut;
memcpy(&ncb.ncb_name,NameString,lstrlen(NameString));
fMain->Refresh();
ncb.ncb_buffer=(PUCHAR)Buffer;
ncb.ncb_length=Volum;
memcpy(&ncb.ncb_callname,PriemString,lstrlen(PriemString));
Netbios(&ncb);
    NBNAMENUM=ncb.ncb_num;
    count=0;
    Begin=clock();

    while(1){
     char Buffer[B_S]={0};
     count++;
     if (count>(s/Volum))
     {
      if (Volum == B_S)
      {
       Buffer[B_S]=NULL;
       Buffer[1]='~';
       Buffer[2]='~';
       Buffer[3]=' ';
ncb.ncb_command=NCBDGSEND;
ncb.ncb_buffer=(PUCHAR)Buffer;
ncb.ncb_length=Volum;//sizeof(Buffer);
Netbios(&ncb);
Memo->Lines->Add("Сообщениебылоотослано ");
Memo->Lines->Add(" ");
       TIMES[i]=Tim;

deltaX=(fMain->Im->Width)/7;
PaintStatistic();
        break;
       }
       else
```

```
          {
           Buffer[B_S]=NULL;
           Buffer[1]='~';
           Buffer[2]=' ';
           Buffer[3]=' ';
  ncb.ncb_command=NCBDGSEND;
  ncb.ncb_buffer=(PUCHAR)Buffer;
  ncb.ncb_length=Volum;
  Netbios(&ncb);
           Memo->Lines->Add("Next: "+IntToStr(Volum*2)+" byte(s)");
           Memo->Lines->Add(" ");
           TIMES[i]=Tim;
           break;
          }
        }
        else
        {
         Buffer[B_S]=NULL;
  inpFile->Seek(Volum*(count-1), soFromBeginning);
  inpFile->Read(Buffer, Volum);
          Memo->Lines->Add(IntToStr(count)+": "+ Buffer);
          Memo->Lines->Add(" ");
  ncb.ncb_command=NCBDGSEND;
  ncb.ncb_buffer=(PUCHAR)Buffer;
  ncb.ncb_length=Volum;
  Netbios(&ncb);
  SendData+=Volum;

        }
        Tim=((clock()-Begin)/CLK_TCK);
  fMain->Refresh();
      } // while(1)
     }
     else
     {
      Memo->Lines->Add("Ошибкапередачи" );
```

```
    Memo->Lines->Add(" ");
   }
i++;
   if (i< 7) fMain->bt4Click(fMain);
   else {
lbV->Caption="Packet:";
i = 0;
   }

}

void __fastcallTfMain::FormCreate(TObject *Sender)
{
Im->Canvas->Brush->Color=clWhite;
Im->Canvas->Rectangle(0,0,Im->Width,Im->Height);
rbPol->Checked = 1;
}

void __fastcallTfMain::btGetLANAClick(TObject *Sender)
{
WorkingLana=StrToInt(edGetLANA->Text);
}
```

Примеры результата выполнения программы показаны на рис. 54 и 55.

Клиент

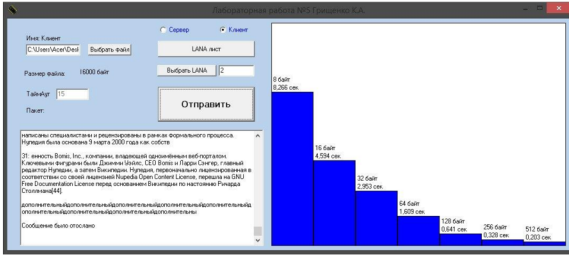

Рис. 54. Результат работы «Клиента»

Сервер

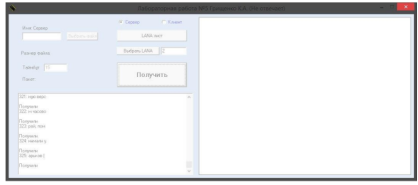

Рис. 55. Результат работы «Сервера»

Задание 3. Тип эксперемента: time out.

Размер пакета 512 байт.

TimeOut в диапозоне от 15 до 25 мс.

Вид соединения: логический канал.

Размер данных: 32 кбайта.

Вывод статистики: на клиенте.

Листинг программы. Реализация на языке Kotlin с использованием GUIJavaFX

ClientMain.kt

```
import javafx.application.Application
import javafx.fxml.FXMLLoader
import javafx.scene.Parent
import javafx.scene.Scene
import javafx.stage.Stage
class ClientMain : Application() {
@Throws(Exception::class)
override fun start(primaryStage: Stage) {
System.setProperty("prism.lcdtext", "false")
val root = FXMLLoad-
er.load<Parent>(javaClass.getResource("sample.fxml"))
primaryStage.title = "Client"
primaryStage.scene = Scene(root, 800.0, 400.0)
primaryStage.show()
}
companion object {
```

```
@JvmStatic
fun main(args: Array<String>) {
launch(ClientMain::class.java)
}
}
}
```

sample.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.chart.CategoryAxis?>
<?import javafx.scene.chart.LineChart?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ClientController">
<center>
<LineChart fx:id="lineChart" alternativeColumnFillVisible="true" pref-Height="340.0" prefWidth="800.0" BorderPane.alignment="CENTER">
<xAxis>
<CategoryAxis fx:id="x" side="BOTTOM" />
</xAxis>
<yAxis>
<NumberAxis fx:id="y" side="LEFT" />
</yAxis>
</LineChart>
</center>
<bottom>
```

```
<Button fx:id="startButton" mnemonicParsing="false" text="Начать передачу" BorderPane.alignment="CENTER" />
</bottom>
<top>
<GridPane BorderPane.alignment="CENTER">
<columnConstraints>
<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
</columnConstraints>
<rowConstraints>
<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
</rowConstraints>
<children>
<Button fx:id="refreshButton" mnemonicParsing="false" text="Обновить текущий IP Адрес" GridPane.columnIndex="1" />
<Label fx:id="currentIpLabel" text="Текущий IP Адресс:">
<padding>
<Insets left="20.0" />
</padding></Label>
<TextField fx:id="serverIpField" promptText="IP Сервера" GridPane.rowIndex="1">
<GridPane.margin>
<Insets left="20.0" right="20.0" />
</GridPane.margin></TextField>
<Label fx:id="totalTimeLabel" GridPane.rowIndex="1" GridPane.columnIndex="1"/>
</children>
</GridPane>
</top>
<padding>
<Insets bottom="5.0" />
```

```
</padding>
</BorderPane>
```

Controller.kt

```
import javafx.fxml.FXML
import javafx.scene.chart.CategoryAxis
import javafx.scene.chart.LineChart
import javafx.scene.chart.NumberAxis
import javafx.scene.chart.XYChart
import javafx.scene.control.Button
import javafx.scene.control.Label
import javafx.scene.control.TextField
import java.io.*
import java.net.NetworkInterface
import java.net.Socket
import java.nio.ByteBuffer
import java.util.*
import java.util.regex.Pattern
import kotlin.collections.ArrayList
import kotlin.collections.HashMap
class Controller {
@FXML
lateinit var startButton: Button
@FXML
lateinit var refreshButton: Button
@FXML
lateinit var currentIpLabel: Label
@FXML
lateinit var totalTimeLabel: Label
@FXML
lateinit var serverIpField: TextField
@FXML
lateinit var lineChart: LineChart<Any, Any>
@FXML
lateinit var x: CategoryAxis
@FXML
lateinit var y: NumberAxis
```

```
        private var socket: SocketThread? = null
        class Graph constructor(val lineChart: LineChart<Any, Any>, val to-
talTimeLabel: Label) {
        private var totalTime = 0L
        private val data: ArrayList<XYChart.Data<Any, Any>> = ArrayList()
        fun addPoint(x: Long, y: Long) {
        val series = XYChart.Series<Any, Any>()
        val point = XYChart.Data<Any, Any>(x.toString(), y)
        data.add(point)
        series.data.addAll(point)
        totalTime += y
        javafx.application.Platform.runLater {
        if (!lineChart.data.isEmpty()) {
        lineChart.data[0].data.add(point)
        } else {
        lineChart.data.add(series)
        }
        totalTimeLabel.text = "Общее время отправки: $y мс"
        }
        }
        fun draw() {
        javafx.application.Platform.runLater {
        totalTimeLabel.text = "Общее время отправки всех пакетов: $totalTime
мс"
        }
        }
        fun clear() {
        data.clear()
        if (!lineChart.data.isEmpty())
        lineChart.data.remove(lineChart.data[0])
        totalTimeLabel.text = ""
        }
        }
        lateinit var graph: Graph
        fun initialize() {
        startButton.setOnAction {
```

```
if (!serverIpField.text.isEmpty() && isIpAddress(serverIpField.text)) {
try {
socket = SocketThread(serverIpField.text, startButton, graph)
} catch (e: Exception) {
e.printStackTrace()
}
socket?.start()
if (socket != null) {
graph.clear()
startButton.isDisable = true
}
}
}
graph = Graph(lineChart, totalTimeLabel)
currentIpLabel.text = "Текущий IP-адрес: " + getCurrentIp()
refreshButton.setOnAction {
currentIpLabel.text = "Текущий IP-адрес: " + getCurrentIp()
}
serverIpField.text = getCurrentIp().substring(0, getCurrentIp().length - 2)
totalTimeLabel.text = "Общее время отправки:"
}
private fun getCurrentIp(): String {
val networkInterfaces = NetworkInterface.getNetworkInterfaces()
while (networkInterfaces.hasMoreElements()) {
val networkInterface = networkInterfaces.nextElement()
val addresses = networkInterface.inetAddresses
while (addresses.hasMoreElements()) {
val address = addresses.nextElement()
if (address.hostAddress != "127.0.1.1"&& address.hostAddress !=
"127.0.0.1"
&& isIpAddress(address.hostAddress)) {
return address.hostAddress
}
}
}
return "Устройство не подключено"
```

```
}
fun isIpAddress(address: String): Boolean {
val pattern = Pattern.compile("^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$")
return pattern.matcher(address).matches()
}
private class SocketThread internal constructor(address: String, val button: Button, val graph: Graph) : Thread() {
val PACKET_SIZE = 512
val DATE_SIZE = 32 * 1024
val MIN_TIME_WAIT = 15
val MAX_TIME_WAIT = 25
val SOCKET_NUMBER = 9999
private val socket: Socket
val input: InputStream
private val output: OutputStream
private val fileInput: FileInputStream
private val buffer = ByteArray(PACKET_SIZE)
init {
socket = Socket(address, SOCKET_NUMBER)
socket.soTimeout = 5 * 1000
input = socket.getInputStream()
output = socket.getOutputStream()
fileInput = FileInputStream("data.txt")
}
private val delays: LinkedList<Int> = LinkedList()
override fun run() {
try {
var isEnd = false
Thread {
try {
var receive = true
while (receive) {
val buf = ByteArray(java.lang.Long.BYTES)
input.read(buf)
val delayReal = bytesToLong(buf)
```

```
if (delayReal == -1L) {
receive = false
} else if (delayReal == -2L) {
continue
}
while (delays.isEmpty()) {
}
val delay = delays.removeFirst()
println("Данныеобработаныза $delayReal мс" +
"при задержке $delay мс")
graph.addPoint(delay.toLong(), delayReal)
}
output.write("stop".toByteArray())
println("вышел")
output.flush()
input.close()
socket.close()
} catch (e: Exception) {
e.printStackTrace()
}
print("Ввод закончен")
isEnd = true
graph.draw()
}.start()
for (i in 0..(MAX_TIME_WAIT - MIN_TIME_WAIT)) {
val delay = MIN_TIME_WAIT + i
for (j in 0..DATE_SIZE / PACKET_SIZE) {
fileInput.read(buffer)
output.write(buffer)
output.flush()
Thread.sleep((delay).toLong())
}
delays.addLast(delay)
buffer[0] = '0'.toByte()
buffer[1] = 'n'.toByte()
buffer[2] = 'e'.toByte()
```

```
buffer[3] = 'x'.toByte()
buffer[4] = 't'.toByte()
buffer.fill('0'.toByte(), 5)
output.write(buffer)
output.flush()
fileInput.channel.position(0)
println("Данные с задержкой $delay мс между пакетами отправлены")
}
output.write("exit".toByteArray())
output.flush()
button.isDisable = false
while (!isEnd) {
}
} catch (e: Exception) {
e.printStackTrace()
button.isDisable = false
}
}
fun bytesToLong(bytes: ByteArray): Long {
val buffer = ByteBuffer.allocate(java.lang.Long.BYTES)
buffer.put(bytes)
buffer.flip()
return buffer.long
}
}
}
```

ServerMain.kt

```
import javafx.application.Application
import javafx.fxml.FXMLLoader
import javafx.scene.Parent
import javafx.scene.Scene
import javafx.stage.Stage
import java.io.BufferedReader
import java.io.IOException
import java.io.InputStreamReader
import java.io.PrintWriter
```

```
import java.net.ServerSocket
import java.net.Socket
/**
* Created by medium on 31.05.17.
*/
class ServerMain: Application() {
override fun start(primaryStage: Stage) {
System.setProperty("prism.lcdtext", "false")
val root = FXMLLoader.load<Parent>(javaClass.getResource
("main.fxml"))
primaryStage.title = "Server"
primaryStage.scene = Scene(root, 800.0, 400.0)
primaryStage.show()
}
companion object {
@JvmStatic
fun main(args: Array<String>) {
launch(ServerMain::class.java)
}
}
}
```

main.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="800.0"
xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="ServerController">
<center>
```

```
<TextArea fx:id="logArea" editable="false" prefHeight="325.0" pref-
Width="800.0" wrapText="true" BorderPane.alignment="CENTER" />
</center>
<top>
<GridPane xmlns="http://javafx.com/javafx/8.0.112"
xmlns:fx="http://javafx.com/fxml/1">
<columnConstraints>
<ColumnConstraints hgrow="SOMETIMES" maxWidth="584.0" min-
Width="10.0" prefWidth="400.0" />
<ColumnConstraints maxWidth="439.0" minWidth="200.0" pref-
Width="400.0" />
<ColumnConstraints maxWidth="439.0" minWidth="200.0" pref-
Width="400.0" />
</columnConstraints>
<rowConstraints>
<RowConstraints minHeight="10.0" prefHeight="30.0"
vgrow="SOMETIMES" />
</rowConstraints>
<Button fx:id="refreshButton" mnemonicParsing="false" text="Обновить
текущий IP Адрес" GridPane.columnIndex="1" />
<Label fx:id="currentIpLabel" text="Текущий IP-Адресс:">
<padding>
<Insets left="20.0" />
</padding>
</Label>
<Button fx:id="rebootButton" mnemonicParsing="false"
text="Перезапустить сервер" GridPane.columnIndex="2" />
<padding>
<Insets bottom="10.0" top="10.0" />
</padding>
</GridPane>
</top>
</BorderPane>
```

ServerController.kt

```
import javafx.beans.value.ChangeListener
import javafx.fxml.FXML
```

```
import javafx.scene.control.Button
import javafx.scene.control.Label
import javafx.scene.control.TextArea
import java.net.NetworkInterface
import java.net.ServerSocket
import java.net.Socket
import java.util.regex.Pattern
import javafx.beans.value.ObservableValue
import java.io.*
import java.nio.ByteBuffer
/**
* Created by medium on 31.05.17.
*/
class ServerController {
private val SOCKET_NUMBER = 9999
private val PACKET_SIZE = 512
@FXML
lateinit var refreshButton: Button
@FXML
lateinit var rebootButton: Button
@FXML
lateinit var currentIpLabel: Label
@FXML
lateinit var logArea: TextArea
val threadRunnable: Runnable = Runnable {
logArea.appendText("Сервер запущен\n")
var out: OutputStream? = null
var servers: ServerSocket? = null
var fromclient: Socket? = null
javafx.application.Platform.runLater {
rebootButton.isDisable = true
rebootButton.text = "Сервер запущен"
}
try {
servers = ServerSocket(SOCKET_NUMBER)
} catch (e: IOException) {
```

```
logArea.appendText("Невозможно прослушать порт " +
SOCKET_NUMBER + "\n")
}
try {
logArea.appendText("Ожидание подключения клиента...\n")
fromclient = servers!!.accept()
logArea.appendText("Клиент " + fromclient.inetAddress.hostAddress + "
подключился\n")
} catch (e: IOException) {
logArea.appendText("Невозможно подключить клиента\n")
}
var `in` = fromclient!!.getInputStream()
var buffer = ByteArray(PACKET_SIZE)
out = fromclient.getOutputStream()
var now = System.currentTimeMillis()
var input = `in`.read(buffer)
now = System.currentTimeMillis() - now
var totalTime = 0L
var time = now
val output: String
var past: String? = null
while ((input) != null) {
var str = String(buffer)
if (str.contains("exit")|| past == str) {
break
} else if (str.contains("0next")) {
out.write(longToBytes(time - now))
out.flush()
println("Данные обработаны за ${time - now} мс")
str = "\nДанные обработаны за ${time - now} мс\n\n"
totalTime+=time
time = 0
} else {
out.write(longToBytes(-2L))
out.flush()
//                println("Пакет обработан за $now мс")
```

```
        str += "\nПакет обработан за $now мс\n"
        }
        javafx.application.Platform.runLater {
        logArea.appendText(str)
        }
        past = str
        now = System.currentTimeMillis()
        input = `in`.read(buffer)
        now = System.currentTimeMillis() - now
        time += now
        }
        out.write(longToBytes(-1L))
        out.flush()
        javafx.application.Platform.runLater {
        rebootButton.isDisable = false
        rebootButton.text = "Перезапустить сервер"
        }
        out.close()
        `in`.close()
        fromclient.close()
        servers?.close()
        javafx.application.Platform.runLater {
        logArea.appendText("Общее время обработки пакетов $totalTime
мс\n")
        }
        }
        fun initialize() {
        currentIpLabel.text = "Текущий IP адрес: " + getCurrentIp()
        refreshButton.setOnAction {
        currentIpLabel.text = "Текущий IP адрес: " + getCurrentIp()
        }
        logArea.textProperty().addListener { observable, oldValue, newValue ->
        logArea.scrollTop = java.lang.Double.MAX_VALUE //this will scroll to
the bottom
        //use Double.MIN_VALUE to scroll to the top
        }
```

```
        rebootButton.setOnAction {
        logArea.text = ""
        startReceive()
        }
        startReceive()
        }
        private fun startReceive(){
        Thread(threadRunnable).start()
        }
        private fun getCurrentIp(): String {
        val networkInterfaces = NetworkInterface.getNetworkInterfaces()
        while (networkInterfaces.hasMoreElements()) {
        val networkInterface = networkInterfaces.nextElement()
        val addresses = networkInterface.inetAddresses
        while (addresses.hasMoreElements()) {
        val address = addresses.nextElement()
        if (address.hostAddress != "127.0.1.1"&& address.hostAddress !=
"127.0.0.1"
        && isIpAddress(address.hostAddress)) {
        return address.hostAddress
        }
        }
        }
        return "Устройство не подключено"
        }
        fun isIpAddress(address: String): Boolean {
        val pattern = Pattern.compile("^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-
9]|25[0-5])\\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$")
        return pattern.matcher(address).matches()
        }
        fun longToBytes(x: Long): ByteArray {
        val buffer = ByteBuffer.allocate(java.lang.Long.BYTES)
        buffer.putLong(x)
        return buffer.array()
        }
        fun bytesToLong(bytes: ByteArray): Long {
```

```
val buffer = ByteBuffer.allocate(java.lang.Long.BYTES)
buffer.put(bytes)
buffer.flip()
return buffer.long
}
```

Примеры результата выполнения программы «окно клиента» после проведения эксперимента показаны на рис. 56 и 57.

Рис. 56. График результата на «Клиенте»

Рис. 57. График результата на «Клиенте»

Пожелания программистам

Чтобы освоить программирование приложений на выше рассмотренных протоколах, необходимо иметь опыт программирования не меньше 1 года, а чтобы правильно уметь ими пользоваться – как минимум 2–3 года. Для начинающих программистов достаточно изучить основные типы библиотеки VCL (хотя некоторые классы в ней не продуманны и не согласованы). Функциональность и замудренность сетевых классов оставляет желать лучшего, а полезны только классы TIdTCPServer, TIdClient, TIdUDPServer и TIdUDPClient. Все остальные сетевые классы нужны, наверное, только самим разработчикам этой библиотеки.

Внимание!!! При сдаче лабораторных работ **ЗАПРЕЩЕНО** использовать библиотеку нестандартных типов AlternativeTypes, так как она содержит сетевые классы с автоматическим управлением каналами и протоколами, а значит, нет смысла изучать устройство сокетов. Разрешается только ознакомиться со всеми возможностями этих сетевых протоколов и изучить их свойства. Также запрещено использование в частных и коммерческих целях без уведомления автора.

В данном пособии рассмотрено 30 % материала по управлению этими протоколами, другие 30 % вы узнаете из лабораторных, а 40 % информации вы уже должны знать, иначе нет смысла даже пытаться программировать с нуля.

САМОЕ ГЛАВНОЕ для программиста – это красивый почерк в его программном коде.

Для решения задач, связанных с написанием программ для организации обмена данными с сети с использованием сетевых протоколов, в качестве эффективного инструментального средства можно воспользоваться модулем Alternative Types Library.

4.4. Краткое описание модуля Alternative Types Library

Модуль Alternative Types Library содержит множество типов данных, позволяющих сократить программный код и время написания программы. Содержит часто пополняющуюся библиотеку постоянно используемых типов, реализованных в виде классов. Позволяет создавать сложные мультимедиа - и графические программы. Является средой разработки серверных и клиентских сетевых приложений. Имеет возможность доступа ко многим системным функциям и устройствам компьютера.

Основная задача этого модуля состоит в быстром написании сложных программных продуктов и избегание ошибок в сложных операциях вычисления и работы с памятью. Все типы имеют автоматическое выделение и освобождение памяти, открытие и закрытие хэндлов, подключение и отключение устройств. Многие типы связаны совместным использованием и зависят друг от друга, что позволяет избежать преобразований внутри классов и накопления одинакового кода.

4.4.1. Список типов в модуле версии (12–11–2017)

4.4.1.1. Базовые типы

ATCharString – строка символов.
ATStringList – список строк.
ATStringArray – массив строк.
ATData – блок памяти.

4.4.1.2. Главные типы

ATThread – создание и управление нитью.
ATCriticalSection – критическая секция.
ATBinaryFile – доступ и управление файлами.

4.4.1.3. Дополнительные типы

ATCallbackService – получение сообщений событий.
ATGraphicsDC – управление графической информацией.

4.4.1.4. Стандартные типы

ATRegistryKey – доступ к ключам реестра.
ATTimer – управление вызовом функции через интервал времени.
ATTime – управление системным временем.
ATWindow – управление окнами.
ATBitmap – графические изображения.
ATDisplay – управление видеоадаптером.
ATResource – доступ к ресурсам исполняемого файла.
ATTrayIcon – управление иконкой в системной панели.
ATPopupMenu – всплывающее оконное меню.
ATCursor – управление курсором мыши.

ATToolHelpSnapshot – снимок памяти процесса.
ATProcess – управление процессами системы.
ATHook – перехват системных сообщений.
ATPipeServer – сервер управления «трубами».
ATPipeClient – клиент управления «трубами».

4.4.1.5. Диалоговые типы

ATOpenDialog – диалог открытия файла.
ATSaveDialog – диалог сохранения файла.
ATColorDialog – диалог выбора цвета.
ATFontDialog – диалог выбора шрифта.

4.4.1.6. Мультимедиа типы

ATMixer – управление звуком.
ATPlayerMCI – проигрыватель звуковых и видеофайлов.
ATPlayerMIDI – синтезатор звуков MIDI.
ATJoystick – управление джойстиком.

4.4.1.7. Сетевые типы

ATNetServerTCP – сервер сетевого IP- протокола TCP- пакетов.
ATNetClientTCP – клиент сетевого IP -протокола TCP -пакетов.
ATNetServerUDP – сервер сетевого IP- протокола UDP- пакетов.
ATNetClientUDP – клиент сетевого IP- протокола UDP- пакетов.
ATNetServerSPX – сервер сетевого IPX- протокола SPX- пакетов.
ATNetClientSPX – клиент сетевого IPX- протокола SPX -пакетов.
ATNetServerIPX – сервер сетевого IPX- протокола IPX- пакетов.
ATNetClientIPX – клиент сетевого IPX- протокола IPX- пакетов.
ATNetbios – управление сетевым протоколом NetBIOS.
ATInternetFile – доступ к файлам HTTP- сервера.
ATNetTunnelTCP – управляемый туннель сетевого IP- протокола TCP-пакетов.

4.4.1.8. Специальные типы

ATCoder – кодирование информации.
ATCompression – сжатие информации.

ATProcessDDE – динамический обмен информацией в процессах.

ATApplication – управление приложением.

4.4.2. Описание базовых типов

4.4.2.1. Описание ATCharString

Краткое описание типа ATCharString

Класс типа ATCharString (строка символов) содержит строку типа char* и функции для её обработки и преобразований. Если вы уже знакомы с типом AnsiString, то вам будут понятны почти все функции ATCharString, а некоторые функции содержат больше параметров и работают быстрее. Требуется при работе почти всех типов Alternative Types Library.

Краткое описание функций и переменных типа ATCharString

ATCharString() – конструктор без параметров.

ATCharString(const int P_Int) – конструктор с преобразованием числа из типа int в строку текущего типа.

ATCharString(const unsigned int P_UnsignedInt) – конструктор с преобразованием числа из типа unsigned int в строку текущего типа.

ATCharString(const double P_Double) – конструктор с преобразованием числа из типа double в строку текущего типа.

ATCharString(const char P_Char) – конструктор с преобразованием символа из типа char в строку текущего типа.

ATCharString(const char* P_String) – конструктор с преобразованием строки из типа char* в строку текущего типа.

ATCharString(const ATCharString& P_CharString) – конструктор с получением строки из адреса переменной P_CharString.

ATCharString(const void* P_Handle) – конструктор с преобразованием адреса из типа void* в строку текущего типа.

~ATCharString() – деструктор строки текущего типа.

char& operator[](const int P_Index) – функция, возвращающая адрес символа строки текущего типа на позиции указанной в переменной P_Index.

char operator[](const int P_Index)const – функция, возвращающая символ строки текущего типа на позиции указанной в переменной P_Index.

void operator=(const ATCharString& P_CharString) – функция, копирующая строку из адреса переменной P_CharString в строку текущего типа.

ATCharString operator+(const ATCharString& P_CharString)const – функция, возвращающая сумму строки текущего типа и строки из адреса переменной P_CharString.

void operator+=(const ATCharString& P_CharString) – функция суммирования строки текущего типа и строки из адреса переменной P_CharString с сохранением в текущем типе.

bool operator==(const ATCharString& P_CharString)const – функция, возвращающая значение **true** при равенстве строки текущего типа и строки из адреса переменной P_CharString, **false** в остальных случаях.

bool operator!=(const ATCharString& P_CharString)const – функция, возвращающая значение **true** при неравенстве строки текущего типа и строки из адреса переменной P_CharString, **false** в остальных случаях.

bool operator>(const ATCharString& P_CharString)const – функция, возвращающая значение **true,** если значение строки текущего типа выше значения строки из адреса переменной P_CharString, **false** в остальных случаях.

bool operator>=(const ATCharString& P_CharString)const – функция, возвращающая значение **true,** если значение строки текущего типа выше или равно значению строки из адреса переменной P_CharString, **false** в остальных случаях.

bool operator<(const ATCharString& P_CharString)const – функция, возвращающая значение **true,** если значение строки текущего типа ниже значения строки из адреса переменной P_CharString, **false** в остальных случаях.

bool operator<=(const ATCharString& P_CharString)const – функция, возвращающая значение **true** если значение строки текущего типа ниже или равно значению строки из адреса переменной P_CharString, **false** в остальных случаях.

void Insert(const ATCharString& P_CharString, const int P_Index) – функция вставки строки из адреса переменной P_CharString в строку текущего типа, начиная с позиции P_Index.

void Delete(const int P_Index, const int P_Length) – функция удаления строки из текущего типа, начиная с позиции P_Index длиной P_Length.

ATCharString SubString(const int P_Index, const int P_Length)const – функция возвращает строку из текущего типа, начиная с позиции P_Index длиной P_Length.

int FindSubString(const int P_Index, const ATCharString& P_CharString)const – функция возвращает индекс строки из адреса переменной P_CharString в строке текущего типа, начиная с позиции P_Index длиной P_Length.

ATCharString FindSeparatedString(int& P_Index, const ATCharString& P_Separator)const – функция возвращает строку из текущего типа, начиная с позиции P_Index до разделительной строки из адреса переменной P_Separator не включительно.

ATCharString RadixToRadix(const int P_RadixNow, const int P_RadixWill)const – функция возвращает строку текущего типа в виде преобразованного числа из P_RadixNow системы измерения в P_RadixWill систему измерения.

int ToInt – переменная содержащая строку текущего типа в виде числа типа int.

double ToDouble – переменная, содержащая строку текущего типа в виде числа типа double.

void* ToHandle – переменная, содержащая строку текущего типа в виде адреса типа void*.

int Length – переменная, содержащая значение длины строки текущего типа и принимающая значения новой длины строки текущего типа.

char* c_str – переменная, содержащая адрес строки текущего типа.

4.4.2.2. Описание ATStringList

Краткое описание типа ATStringList

Класс типа ATStringList (список строк) содержит список ATCharString строк и функции для их обработки и преобразований. Является сокращенным аналогом типа TStringList. Требуется при работе некоторых типов Alternative Types Library.

Краткое описание функций и переменных типа ATStringList

ATStringList() – конструктор без параметров;

ATStringList(const ATCharString& P_CharString) – конструктор с преобразованием строки из адреса переменной P_CharString в список текущего типа.

ATStringList(const ATStringList& P_StringList) – конструктор с получением списка из адреса переменной P_StringList.

~ATStringList() – деструктор списка текущего типа.

ATCharString& operator[](const int P_Index) – функция, возвращающая адрес строки списка текущего типа на позиции, указанной в переменной P_Index.

ATCharString operator[](const int P_Index)const – функция, возвращающая строку списка текущего типа на позиции, указанной в переменной P_Index.

void operator=(const ATStringList& P_StringList) – функция, копирующая список из адреса переменной P_StringList в список текущего типа.

ATStringList operator+(const ATStringList& P_StringList)const – функция, возвращающая сумму списка текущего типа и списка из адреса переменной P_StringList.

void operator+=(const ATStringList& P_StringList) – функция суммирования списка текущего типа и списка из адреса переменной P_StringList с сохранением в текущем типе.

void Add(const ATCharString& P_CharString) – функция добавления строки из адреса переменной P_CharString к списку текущего типа.

void Update(const ATCharString& P_CharString) – функция вставки строки из адреса переменной P_CharString в список текущего типа с сортировкой по возрастанию.

void Insert(const int P_Index, const ATCharString& P_CharString) – функция вставки строки из адреса переменной P_CharString в список текущего типа на позицию P_Index.

void Delete(const int P_Index) – функция удаления строки в списке текущего типа на позиции P_Index.

void Clear() – функция удаления всех строк в списке текущего типа.

int IndexOf(const ATCharString& P_CharString)const – функция возвращает индекс строки из адреса переменной P_CharString в списке текущего типа.

void Move(const int P_CurIndex, const int P_NewIndex) – функция смещает строку с индексом P_CurIndex на позицию P_NewIndex в списке текущего типа.

bool LoadFromFile(const ATCharString& P_FileName) – функция возвращает значение **true,** если произведена успешная запись списка текущего типа в файл с именем, хранящимся в строке из адреса переменной P_FileName, **false** в остальных случаях.

bool SaveToFile(const ATCharString& P_FileName) – функция возвращает значение **true** ,если произведено успешное чтение в список текущего типа из файла с именем, хранящимся в строке из адреса переменной P_FileName, **false** в остальных случаях.

int Count – переменная, содержащая количество строк в списке текущего типа.

ATCharString Strings[const int P_Index] – переменная, содержащая строку списка текущего типа на позиции, указанной в переменной P_Index.

ATCharString Text – переменная, содержащая все строки списка текущего типа, разделенные символами “\r\n”.

4.4.2.3. Описание ATStringArray

Краткое описание типа ATStringArray

Класс типа ATStringArray (массив строк) содержит массив ATCharString строк и функции для их обработки и преобразований. Используется для быстрого доступа к строкам.

Краткое описание функций и переменных типа ATStringArray

Все функции и переменные этого типа эквивалентны функциям и переменным типа ATStringList по действию и имеют одинаковое описание.

4.4.2.4. Описание ATData

Краткое описание типа ATData

Класс типа ATData (память) содержит блоки памяти и функции для их обработки и преобразований, а так же позволяет сократить программный код и избежать трудностей использования операций над памятью.

Краткое описание функций и переменных типа ATData

ATData() – конструктор без параметров.

ATData(const unsigned int P_Size) – конструктор с выделением памяти размером P_Size.

ATData(const ATData& P_Data) – конструктор с получением памяти из адреса переменной P_Data.

ATData(const char* P_String) – конструктор с преобразованием строки из адреса переменной P_String в память текущего типа.

ATData(const void* P_Data, const unsigned int P_Size) – конструктор с получением памяти из адреса переменной P_Data и размером P_Size.

~ATData() – деструктор памяти текущего типа.

void operator=(const ATData& P_Data) – функция получения памяти из адреса переменной P_Data в память текущего типа.

void Load(const void* P_Data, const unsigned int P_Size) – функция получения памяти из адреса переменной P_Data размером P_Size со стиранием текущей.

ATData SubData(const unsigned int P_Index, const unsigned int P_Size) – функция возвращает память текущего типа, начиная с позиции P_Index длиной P_Size.

void Write(const unsigned int P_Index, const void* P_Data, const unsigned int P_Size) – функция записи памяти из адреса переменной P_Data длиной P_Size в память текущего типа, начиная с позиции P_Index.

void Add(const void* P_Data, const unsigned int P_Size) – функция добавления памяти из адреса переменной P_Data длиной P_Size в память текущего типа.

void Insert(const unsigned int P_Index, const ATData& P_Data) – функция вставки памяти из адреса переменной P_Data в память текущего типа, начиная с позиции P_Index.

void Delete(const unsigned int P_Index, const unsigned int P_Size) – функция удаления памяти текущего типа, начиная с позиции P_Index длиной P_Size.

bool FindData(unsigned int& P_Index, const ATData& P_Data) – функция поиска фрагмента памяти из переменной P_Data в памяти текущего типа, начиная с позиции P_Index с записью результата в него.

void Zero() – функция преобразования всей памяти текущего типа в значение 0.

void Clear() – функция удаления всей памяти текущего типа.

void* Data – переменная, содержащая адрес всей памяти текущего типа и принимающая значение нового адреса текущего типа (только для совместимости с некоторыми функциями записи!).

unsigned int Size – переменная, содержащая размер всей памяти текущего типа и принимающая значение нового размера текущего типа.

unsigned short CheckSum – переменная, содержащая значение суммы байтов всей памяти текущего типа.

ЗАКЛЮЧЕНИЕ

В данном учебном пособии автор систематизировал материал области сетевых технологий, который является фундаментальной базисной основой для понимания основных моментов, связанных с основами построения и эксплуатации компьютерных сетей. В изложенном материале показана связь дисциплины с другими общеобразовательными и профилирующими дисциплинами, изучаемыми студентами в рамках направлений 09.03.04, 02.03.03, такими как «Информатика», «Основы алгоритмизации и программирования», «Операционные системы» (ОС), «Администрирование информационных систем», а также теория сигналов, кодирование. Из этого следует, что дисциплина как бы интегрирует в себе вышеуказанные направления, в чем можно убедиться на примере широты спектра терминов ее предметной области.

В процессе чтения курса лекций по данной дисциплине можно сделать выводы, ч то, несмотря на знакомство с основами технологии работы в сети, уже на младших курсах в процессе изучения такого предмета, как «Информатика», у студентов возникают определенные трудности с усвоением изучаемого материала вследствие терминологической насыщенности, трудности понимания процессов, протекающих в сетях, динамики развития данного направления.

Поэтому основной целью данного пособия, как было указано выше, является: стремление улучшить доступность материала, увеличить полноту материала, упростить восприятие, отразить основные тенденции в развитии данного направления в целом.

В дальнейшем в плане развития и модернизации данной дисциплины планируется издание учебно-методических пособий, в которых будут отражены основные моменты, связанные с изучением протоколов взаимодействия и методов доступа в современных КС различного уровня и ранга, удаленных коммуникаций на базе модемов, сетевого оборудования и организации КС, современных сетевых операционных систем.

Надеемся, что пособие позволит улучшить качество изучения материала по данной дисциплине и тем самым повысить качество знаний.

СПИСОК ЛИТЕРАТУРЫ

1. *Блэк, Ю.* Сети ЭВМ: протоколы,стандарты,интерфейсы [Текст] / Ю. Блэк ; под ред. В. В. Василькова. – М. : Мир, 1990. – 510 с.
2. *Мартин, Дж.* Введение в сетевые технологии. Практическое руководство по организации сетей [Текст] / Дж. Мартин. – М. : Лори, 2002. – 659 с.
3. *Таненбаум, Э.* Компьютерные сети [Текст] / Э. Таненбаум, Д. Уэзеролл. – СПб. : Питер, 2013. – 944 с.
4. *Пятибратов, А. П.* Вычислительные системы, сети и теле- коммуникации [Текст] / А. П. Пятибратов, А. А. Кириченко, Л. П. Гудыно. – М. : Финансы и статистика, 2008. – 736 с.
5. *Пятибратов, А. П.* Вычислительные системы, сети и телекоммуникации : учебное пособие для вузов [Текст] / А. П. Пятибратов, А. А. Кириченко, Л. П. Гудыно. – М. : КноРус, 2013. – 372 с.
6. Основы локальных сетей: курс лекций. учебное пособие [Текст] / В. Ю. Новиков, С. В. Кондратенко. – М. : ИНТУИТ.РУ «Интернет-университет Информационных технологий», 2009. – 360 с.
7. *Бертсекас, Д.* Сети передачи данных [Текст] / Д. Бертсекас, Р. Галлагер ; под ред. Б. С. Цыбакова. – М. : Мир, 1989. – 544 с.
8. Основы сетей передачи данных: курс лекций : учебное пособие [Текст] / В. Г.Олифер, Н. А. Олифер. – М. : ИНТУИТ.РУ «Интернет-университет Информационных технологий», 2005. – 176 с.
9. *Олифер, В. Г.* Компьютерные сети. Принципы, технологии, протоколы [Текст] / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2013. – 944 с.
10. *Олифер, В. Г.* Компьютерные сети. Принципы, технологии, протоколы [Текст] / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2010. – 944 с.
11. *Олифер В. Г.* Сетевые операционные системы [Текст] / В. Г.Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 544 с.
12. *Таненбаум, Э.* Компьютерные сети [Текст] / Э. Таненбаум. – СПб. : Питер, 2011. – 944 с.
13. *Головин, Ю. А.* Информационные сети [Текст] / Ю. А. Головин, А. А. Суконщиков, С. А. Яковлев. – М. : Academia, 2013. – 384 с.
14. TCP/IP, 2015. – https://ru.wikipedia.org/wiki/TCP/IP.
15. *Столлингс, В.* Компьютерные сети, протоколы и технологии Интернета [Текст] / В. Столлингс. – СПБ. : БХВ-Петербург, 2013. – 832 с.
16. *Фейт, С.* TCP/IP. Архитектура, протоколы, реализация (включая IPv6 и IPSecurity) [Текст] / С. Фейт. – М. : Лори, 2015. – 424 с.
17. *Камер, Д.* Сети TCP/IP. Том 1. Принципы, протоколы и структура [Текст] / Д. Камер. – М. : Вильямс, 2003. – 880 с.
18. *Семенов, Ю. А.* Телекоммуникационные технологии (v 5.1), 2014. – http://book.itep.ru/.
19. *Семенов, Ю. А.* Протоколы и ресурсы Интернет [Текст] / Ю. А. Семенов. – М. : Радио и связь, 2006.

20. *Семенов, Ю. А.* Сети Интернет.Архитектура и протоколы [Текст] / Ю. А. Семенов. – М. : Сириъ, 2008.
21. *Семенов, Ю. А.* Протоколы Интернет. Энциклопедия [Текст] / Ю. А. Семенов. – Горячая линия – Телеком, М. : 2011.
22. *Проскуряков, А. В.* и др. Сети ЭВМ и телекоммуникации. Основы построения вычислительных сетей : учебное пособие по курсу «Сети ЭВМ и телекоммуникации» Ч.1. [Текст] / А. В. Проскуряков. – Таганрог : ТРТУ, 2003. – 112 с.

СПИСОК СОКРАЩЕНИЙ

АИМ – амплитудно-импульсная модуляция;
АКД – аппаратура окончания канала данных;
АМ – амплитудная модуляция;
АПД – аппаратура передачи данных;
АСОД – автоматизированная система обработки данных;
АЧХ – амплитудно-частотная характеристика;
БЭМ ВОС – базовая эталонная модель взаимодействия открытых систем;
ВИМ – времяимпульсная модуляция;
ГВС – глобальная вычислительная сеть;
Гор. ВС – городская вычислительная сеть;
ДЧ – диапазон частот;
ИВС – информационно-вычислительная сеть;
КИМ – кодоимпульсная модуляция;
КПО – комплекс программного обеспечения;
КТС – комплекс технических средств;
КС – компьютерная сеть;
ЛВС – локальная вычислительная сеть;
МОС – международная организация по стандартизации;
ОГВС – общегосударственная вычислительная сеть;
ООД – оконечное оборудование данных;
РВС – региональная вычислительная сеть;
РС – рабочая станция;
СОС – сетевая операционная система;
СрОД – средства обработки данных;
СВЧ – сверхвысокие частоты;
СЧИМ – счетно-импульсная модуляция;
ФС – файловый сервер;
ФИМ – фазоимпульсная модуляция;
ФМ – фазовая модуляция;
ЧИМ – частотно-импульсная модуляция;
ЧМ – частотная модуляция;
ШИМ – широтно-импульсная модуляция;
ЭВМ – электронно-вычислительная машина;
IP – протокол межсетевого взаимодействия, интернет протокол;
IPX – протокол межсетевого обмена пакетами компании Novell;

NetBIOS – протокол сетевой базовой системы ввода-вывода;
SPX – протокол последовательного обмена пакетами компании Novell;
TCP – протокол управления передачей;
UDP – дейтаграммный протокол пользователя;
TCP / IP – промышленный стек протоколов;
IPX / SPX – стек протоколов компании Novell.

СОДЕРЖАНИЕ

Учебное пособие

ПРОСКУРЯКОВ Александр Викторович

КОМПЬЮТЕРНЫЕ СЕТИ. ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ И ТЕЛЕКОММУНИКАЦИЙ

Учебное пособие

Редакторы *И. А. Проценко, З. И. Надточий*
Корректор *Т. Ф. Кочергина*

Сдано в набор 29.12.2017 г.
Бумага офсетная. Печать офсетная. Формат 60×84 $^{1}/_{16}$.
Усл. печ. лист. 11,68. Уч. изд. л. 9,0. Заказ № 6674. Тираж 40 экз.

Издательство Южного федерального университета.

Отпечатано в отделе полиграфической, корпоративной и сувенирной продукции
Издательско-полиграфического комплекса КИБИ МЕДИА ЦЕНТРА ЮФУ.
344090, г. Ростов-на-Дону, пр. Стачки, 200/1, тел (863) 243-41-66.